

TRABALHO DE GRADUAÇÃO

**DESENVOLVIMENTO DE UM SISTEMA DE LOCALIZAÇÃO
PARA ROBÔS MÓVEIS BASEADO EM FILTRAGEM
BAYESIANA NÃO-LINEAR**

Camila Gonçalves de Brito

Brasília, julho de 2017



**ENGENHARIA
MECATRÔNICA**
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

DESENVOLVIMENTO DE UM SISTEMA DE LOCALIZAÇÃO
PARA ROBÔS MÓVEIS BASEADO EM FILTRAGEM
BAYESIANA NÃO-LINEAR

Camila Gonçalves de Brito

*Relatório submetido como requisito parcial de obtenção
de grau de Engenheiro de Controle e Automação*

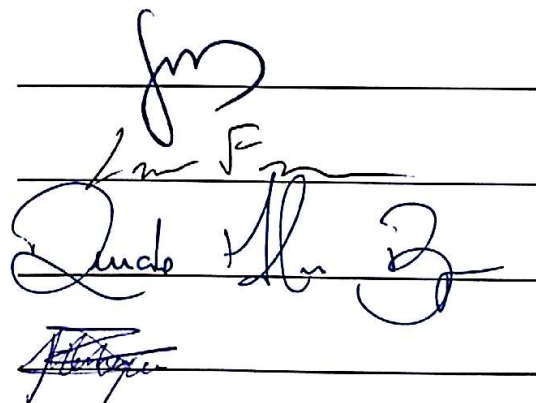
Banca Examinadora

Prof. Geovany Araújo Borges, ENE/UnB
Orientador

Dr. Luis Felipe da Cruz Figueredo, ENE/UnB
Coorientador

Prof. Renato Alves Borges, ENE/UnB
Examinador interno

Prof. Henrique Marra Taira Menegaz,
FGA/UnB
Examinador interno



Brasília, julho de 2017

FICHA CATALOGRÁFICA

DE BRITO, CAMILA GONÇALVES

Desenvolvimento de um Sistema de Localização para Robôs Móveis Baseado em Filtragem Bayesiana Não-Linear,

[Distrito Federal] 2017.

viii, 63p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2017). Trabalho de Graduação – Universidade de Brasília.Faculdade de Tecnologia.

1.Robótica Móvel

2.Navegação Autônoma

3.Fusão Sensorial

4.Filtro de Kalman

I. Mecatrônica/FT/UnB

II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

DE BRITO, C.G., (2017) Desenvolvimento de um Sistema de Localização para Robôs Móveis Baseado em Filtragem Bayesiana Não-Linear. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT. TG-*n*º007, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 63p.

CESSÃO DE DIREITOS

AUTOR: Camila Gonçalves de Brito

TÍTULO DO TRABALHO DE GRADUAÇÃO: Desenvolvimento de um Sistema de Localização para Robôs Móveis Baseado em Filtragem Bayesiana Não-Linear.

GRAU: Engenheiro

ANO: 2017

É concedida à Universidade de Brasília permissão para reproduzir cópias deste Trabalho de Graduação e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte desse Trabalho de Graduação pode ser reproduzida sem autorização por escrito do autor.

Camila Gonçalves de Brito

Campus Darcy Ribeiro, SG-11, Universidade de Brasília.

70919-970 Brasília – DF – Brasil.

Dedicatória

À minha mãe e ao meu pai, que eu amo tanto.

Camila Gonçalves de Brito

Agradecimentos

Gostaria de agradecer em primeiro lugar à minha família maravilhosa, que é minha maior fonte de inspiração, meu porto seguro, meus maiores companheiros. Obrigada pai e mãe por serem os melhores exemplos e por sempre cuidarem de mim com tanto amor. Obrigada irmã, por me ensinar tanta coisa, rir e compartilhar tantos momentos comigo. Amo vocês!

Em segundo lugar, gostaria de agradecer ao meu menino, Johnny, por ser o melhor namorado e estar sempre ao meu lado. Obrigada por acreditar em mim, me incentivar, me fazer rir, cuidar de mim e compartilhar todos os bons e maus momentos. Obrigada por me ajudar a fazer todos os testes e me abraçar nos momentos de frustração. Ter você ao meu lado durante essa jornada é maravilhoso. Te amo!

Obrigada à equipe mais linda de todas, a DROID, e todos os seus membros maravilhosos. Vocês foram a melhor experiência da minha graduação e esse trabalho não existiria sem vocês. Vou levar para sempre todas as experiências, ensinamentos e amizades que fiz na equipe. Um agradecimento especial para todos que trabalharam na criação do Bruce, ao Thiago, ao Luan e aos meus companheiros de TCC Buzz, Let e Bau. Buzz e Let, vocês são pessoas incríveis! Foi um prazer trabalhar com vocês nesse e em tantos outros projetos. Não poderia deixar de agradecer especialmente aos amigos queridos Artur, Ph e Lets.

Gostaria de agradecer também aos amigos mais incríveis que alguém poderia ter, Rê, Tutty e Thomz, por me acompanharem em todos os momentos da vida, desde a época do ensino fundamental. É um privilégio ter passado os últimos 10 anos com vocês! Um obrigado especial e carinhoso ao Dani, por também fazer parte dessa conquista.

Por fim, gostaria de agradecer aos meus professores e mestres, em especial ao professor Geovany e ao Luis, por serem excelentes orientadores e tão atenciosos, sempre me aconselhando para o melhor e sendo grandes inspirações.

Gratidão a todos!

Camila Gonçalves de Brito

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema de localização para robôs móveis em ambientes externos. O sistema é baseado na integração da odometria, de um receptor de GPS e de uma unidade de navegação inercial contendo um giroscópio, acelerômetro e magnetômetro triaxiais. A fusão sensorial é realizada por meio de algoritmos de filtragem Bayesiana não-linear, em especial o Filtro de Kalman Estendido e o Filtro de Kalman *Unscented*. Devido a problemas com a coleta de dados do GPS, foi realizada a estimação da posição e da velocidade do robô apenas por meio da odometria e os filtros foram utilizados na estimação da atitude. Os resultados encontrados validam a abordagem adotada. Este trabalho foi realizado no contexto de competições de robótica baseadas em navegação autônoma e, assim, espera-se que ele contribua como uma base para o desenvolvimento de um sistema de localização robusto para os robôs da equipe DROID.

Palavras Chave: robótica móvel, navegação autônoma, localização, fusão sensorial, filtro de Kalman

ABSTRACT

This work presents the development of a localization system for mobile robots in outdoor environments. The system is based on the integration of odometry information, a GPS receiver and an inertial measurement unit, containing triaxial gyroscope, accelerometer and magnetometer. The sensor fusion is accomplished with the use of non-linear bayesian filtering algorithms, in particular the Extended Kalman Filter and the Unscented Kalman Filter. Due to problems with GPS data collection, the position and velocity of the robot were estimated only by means of odometry and the filters were used to estimate the attitude. The results validate the adopted approach. This work was done in the context of robotics competitions based on autonomous navigation and thus is expected to contribute as a basis for developing robust localization system for the DROID team robots.

Keywords: robotics, mobile robots, autonomous navigation, localization, sensor fusion, kalman filter

SUMÁRIO

1	INTRODUÇÃO	1
1.1	CONTEXTUALIZAÇÃO	1
1.1.1	A <i>RoboMagellan</i>	2
1.2	DEFINIÇÃO DO PROBLEMA	3
1.3	OBJETIVOS DO PROJETO	4
1.4	RESULTADOS OBTIDOS	5
1.5	APRESENTAÇÃO DO MANUSCRITO	5
2	FUNDAMENTOS	6
2.1	TÉCNICAS DE LOCALIZAÇÃO	6
2.1.1	MÉTODOS RELATIVOS	6
2.1.2	MÉTODOS ABSOLUTOS	9
2.1.3	FUSÃO SENSORIAL	11
2.2	FUNDAMENTOS MATEMÁTICOS	12
2.2.1	SISTEMAS DE COORDENADAS	12
2.2.2	ALGORITMO TRIAD	14
2.2.3	FILTRAGEM BAYESIANA	15
3	DESENVOLVIMENTO	19
3.1	ABORDAGEM ADOTADA	19
3.2	MODELO CINEMÁTICO DO ROBÔ	21
3.3	MODELO DO PROCESSO	22
3.4	MODELO DE MEDIÇÃO	23
3.5	EQUAÇÕES PARA EKF E UKF	23
3.6	DESCRIÇÃO DOS COMPONENTES	24
3.6.1	CODIFICADOR INCREMENTAL	24
3.6.2	RECEPTOR DE GPS	27
3.6.3	IMU	28
3.7	IMPLEMENTAÇÃO	28
3.7.1	COLETA DE DADOS	29
3.7.2	IMPLEMENTAÇÃO DOS FILTROS DE KALMAN	32
3.7.3	IMPLEMENTAÇÃO DA ODOMETRIA	32

4	RESULTADOS	35
4.1	TESTES DE COLETA DE DADOS	35
4.1.1	ATRASO DO GPS	35
4.1.2	TAXAS DE AMOSTRAGEM.....	37
4.1.3	PERFORMANCE DO MODELO DE PROCESSO E DO MODELO DE MEDIÇÃO	39
4.1.4	FUSÃO SENSORIAL	39
4.2	TESTE DE LOCALIZAÇÃO POR ODOMETRIA EM TEMPO REAL.....	45
4.3	DIFICULDADES ENCONTRADAS	45
5	CONCLUSÕES	47
5.1	TRABALHOS FUTUROS	47
	REFERÊNCIAS BIBLIOGRÁFICAS	49
	ANEXOS.....	52
I	DESCRIÇÃO DO CONTEÚDO DO CD.....	53
II	FILTROS DE KALMAN	54
II.1	FILTRO DE KALMAN.....	54
II.2	FILTRO DE KALMAN ESTENDIDO.....	55
II.3	FILTRO DE KALMAN UNSCENTED.....	56
II.3.1	ALGORITMO USQUE	57
III	QUATERNIOS	60
III.1	DEFINIÇÃO	60
III.2	PROPRIEDADES.....	60
III.3	REPRESENTAÇÃO DE ATITUDE COM QUATÉRNIOS	61
III.4	PROPAGAÇÃO DE ATITUDE.....	63
III.5	TRANFORMAÇÃO ENTRE ÂNGULO DE EULER E QUATÉRNIO	63

LISTA DE FIGURAS

1.1	<i>Boss</i> , o veículo autônomo que ganhou o DARPA <i>Urban Challenge</i> de 2007.....	2
1.2	Robô navegando durante a <i>RoboMagellan</i>	3
1.3	Robô de navegação autônoma da equipe DROID	4
1.4	Fluxograma da arquitetura do robô da DROID	5
2.1	Método de localização relativa [1]	7
2.2	Esquema simplificado de um acelerômetro [1]	8
2.3	Esquema simplificado de um giroscópio vibratório	9
2.4	Princípio da trilateração	10
2.5	Fontes de erro na propagação de sinal do GPS [2]	11
2.6	Efeito da disposição geométrica dos sinais na acurácia da estimação de posição[1].....	11
2.7	Sistemas de coordenadas usados neste trabalho.....	14
2.8	Comparação entre a amostragem real, a linearização de primeira ordem do EKF e a transformação <i>unscented</i> do UKF [3]	17
3.1	Diagrama de blocos do EKF para localização 3D	20
3.2	Diagrama de blocos do UKF para localização 3D	20
3.3	Parâmetros que descrevem a pose e a cinemática do robô a tração diferencial	21
3.4	GPS NEO-6M da <i>u-blox</i>	27
3.5	IMU MPU-9250 da <i>InvenSense</i>	28
3.6	Evolução dos parâmetros estimados para o acelerômetro	32
3.7	Programas de coleta de dados	33
3.8	Fluxograma da subrotina de processamento das mensagens do GPS.....	34
4.1	Caminhos fechados realizados pelo robô	36
4.2	Pontos fornecidos pelo programa de coleta de dados do GPS durante o percurso da Trajetória 3	37
4.3	Taxa de amostragem dos sensores durante o percurso da Trajetória 1 pelo tempo de execução.....	37
4.4	Taxa de amostragem dos sensores durante o percurso da Trajetória 1 pelo número de amostras	38
4.5	Estimação da pose do robô na Trajetória 1	40
4.6	Estimação da pose do robô na Trajetória 2	41
4.7	Estimação da pose do robô na Trajetória 3	42

4.8	Pose estimada do robô na Trajetória 1 após filtragem	43
4.9	Pose estimada do robô na Trajetória 2 após filtragem	43
4.10	Pose estimada do robô na Trajetória 3 após filtragem	44
4.11	Trajetória real do robô vs trajetória de referência utilizando um sistema de locali- zação em tempo real por odometria	45

LISTA DE TABELAS

2.1	Elementos do campo magnético em Brasília	9
3.1	Performance do GPS <i>NEO-6M</i>	27
3.2	Protocolos disponíveis para o GPS <i>NEO-6M</i>	28
3.3	Características da MPU-9250.....	29
4.1	Distância entre os pontos inicial e final das trajetórias para cada método utilizado ...	44

LISTA DE SÍMBOLOS

Símbolos Latinos

X, Y, Z	Eixos ortonormais de um sistema de coordenadas tridimensional
\mathbf{r}	Vetor de posição
\mathbf{v}	Vetor de velocidade
\mathbf{f}	Vetor de força específica
\mathbf{g}	Vetor de aceleração gravitacional
\mathbf{m}	Vetor de campo magnético
\mathbf{R}_f^m	Matriz de rotação do sistema M para o F
\mathbf{q}_f^m	Quatérnio de rotação do sistema M para o F
r	Raio das rodas do robô
B	Distância entre as rodas do robô
h	Altitude

Símbolos Gregos

ω_{fm}^m	Velocidades angulares do sistema M em relação ao sistema F expresso no sistema M
φ	Ângulo do eixo de tração da roda
ψ	Ângulo de guinada (<i>yaw</i>)
θ	Ângulo de arfagem (<i>pitch</i>)
ϕ	Ângulo de rolagem (<i>roll</i>)
λ	Latitude
Λ	Longitude

Subscritos

e	Valor referente à roda esquerda
d	Valor referente à roda direita
ref	Referência

Sobrescritos

\cdot	Variação temporal
\sim	Valor corrompido por erros
\wedge	Valor estimado
e	Vetor representado no sistema E
n	Vetor representado no sistema N
b	Vetor representado no sistema B

Siglas

DROID	Divisão de Robótica Inteligente
EKF	<i>Extended Kalman Filter</i>
GNSS	<i>Global Navigation Satellite System</i>
GPS	<i>Global Positioning System</i>
I ² C	<i>Inter-Integrated Circuit</i>
IMU	<i>Inertial Measurement Unit</i>
INS	<i>Inertial Navigation System</i>
KF	<i>Kalman Filter</i>
LSB	<i>Least Significant Bit</i>
MEMS	<i>MicroElectroMechanical Systems</i>
NMEA	<i>National Marine Electronics Association</i>
ROS	<i>Robot Operating System</i>
UART	<i>Universal Asynchronous Receiver/Transmitter</i>
UKF	<i>Unscented Kalman Filter</i>
WGS84	<i>World Geodetic System</i>

Capítulo 1

Introdução

1.1 Contextualização

A robótica, desde seu princípio em meados de 1960, era focada principalmente no projeto e controle de manipuladores, motivados pelas necessidades da indústria [4]. Nos dias de hoje, porém, o uso de robôs está cada vez mais presente em nosso cotidiano, visto de diferentes perspectivas. Robôs podem, por exemplo, ser usados para explorar ambientes inacessíveis a humanos, como outros planetas ou o fundo do mar; podem ser usados para realizar reparos e manutenções remotamente; podem ser usados para melhorar a qualidade de vida das pessoas, com o advento de robôs domésticos e carros que se dirigem sozinhos. Em todas as aplicações mencionadas, o robô requer *mobilidade e autonomia*.

A mobilidade diz respeito ao grau com que o robô é capaz de se mover livremente pelo mundo [5]. Já autonomia se refere a capacidade de operar no mundo real sem qualquer forma de controle externo, ou seja, robôs autônomos são aqueles capazes de realizar tarefas sem qualquer interferência humana. [5, 6].

Dessa forma, uma das tarefas mais importantes para robôs móveis autônomos é a *autolocalização*. De acordo com [7, 8], a capacidade de se localizar no mundo é essencial para dar real autonomia ao robô, uma vez que para tomar qualquer decisão corretamente, o robô precisa saber onde está.

Assim, dada a importância do problema da localização e sua extensa área de aplicação, competições de robótica com foco em automação de veículos são um dos maiores incentivos para que novas técnicas e tecnologias sejam desenvolvidas e aplicadas na área. Os maiores exemplos são os desafios promovidos pelo Departamento de Defesa dos Estados Unidos, o DARPA *Grand Challenge*, cujo objetivo era desenvolver veículos autônomos capazes de navegar por trilhas no deserto e rodovias em alta velocidade [9] e o DARPA *Urban Challenge*, cujo objetivo era desenvolver veículos para navegar em um ambiente urbano, interagindo com outros veículos em movimento e obedecendo as leis de trânsito da Califórnia [10]. O ganhador do desafio de 2007 é mostrado na Figura 1.1.

Existem também campeonatos de navegação autônoma em menor escala, destinados a alunos



Figura 1.1: *Boss*, o veículo autônomo que ganhou o DARPA *Urban Challenge* de 2007

de graduação e *hobbyistas*, como a *RoboMagellan* da *Robogames*¹ e a *Robô Trekking* da *Winter Challenge*².

Foi nesse contexto de competições para o desenvolvimento em robótica autônoma que surgiu a equipe DROID - Divisão de Robótica Inteligente - formada por alunos de engenharia da Universidade de Brasília - UnB. A DROID, fundada em 2009, desenvolve e constrói robôs autônomos capazes de resolver desafios inteligentes. Visando expandir seus horizontes e entrar no mundo da navegação autônoma, a equipe decidiu participar da categoria *RoboMagellan* do evento mundial *RoboGames*.

1.1.1 A *RoboMagellan*

A *RoboGames*³ é o maior evento aberto de competições de robótica do mundo, considerada as Olimpíadas da Robótica. Ela reúne inúmeras nacionalidades em mais de 50 categorias de competições, que vão desde guerra de robôs a desafios de robótica autônoma.

A categoria de interesse para a equipe DROID e que este trabalho aborda é a *RoboMagellan*, uma competição que enfatiza a navegação autônoma e desvio de obstáculos em ambiente externo e em terreno variado.

Os robôs participantes devem ser programados para percorrer uma trajetória não-linear a céu aberto, devendo passar por determinados pontos de parada, demarcados por cones de trânsito. O robô conhece, previamente, as coordenadas de GPS de sua posição inicial e de cada um dos pontos de parada. Assim, os robôs devem navegar autonomamente entre os marcos, sendo capazes

¹Regras RoboMagellan, disponível em <http://robogames.net/rules/magellan.php>

²Regras Robo Trekking, disponível em https://www.robocore.net/upload/attachments/robocore__regras_roboto_trekking-100.pdf

³<http://robogames.net>

de andar em diversos terrenos, como pavimento, asfalto e grama, enquanto desviam de obstáculos (árvores, postes, bancos, pessoas, etc). A pontuação é dada de acordo com a acurácia e a velocidade do robô. Um exemplo é mostrado na Figura 1.2.



Figura 1.2: Robô navegando durante a *RoboMagellan*

1.2 Definição do Problema

Este trabalho faz parte de um conjunto de projetos que abordam o desenvolvimento do robô construído pela equipe DROID para participar da *RoboMagellan*. O desafio consiste em dois problemas principais: a navegação autônoma e a detecção de marcos visuais, isto é, dos cones de trânsito. De acordo com Choset H., et al. [11], a navegação é a tarefa de encontrar qual movimento deve ser feito pelo robô para que ele vá de uma configuração (ou estado) para outra, sem colidir. Assim, o projeto do robô foi dividido nas seguintes tarefas:

- Localização, tarefa abordada neste trabalho, na qual o robô deve determinar, com o passar do tempo, onde está no mundo;
- Planejamento de rotas e arquitetura geral de navegação, abordado em [12], na qual o robô, de posse de sua localização e do mapa do ambiente, deve determinar a melhor trajetória para atingir a posição final desejada, evitando obstáculos contidos no mapa;
- Sistema de controle de velocidade e desvio reativo de obstáculos, abordado em [13], na qual, de posse da trajetória a ser realizada, deve-se fazer o controle da velocidade do robô para garantir que a rota será seguida corretamente, bem como desviar de obstáculos não previstos no mapa; e
- Sistema de detecção de marcos, que será abordado em trabalhos futuros, na qual o robô deve, por meio de visão computacional, identificar o cone de trânsito, garantido que ele atingirá exatamente o ponto desejado.

O robô da DROID, mostrado na Figura 1.3, foi inteiramente construído pela equipe.

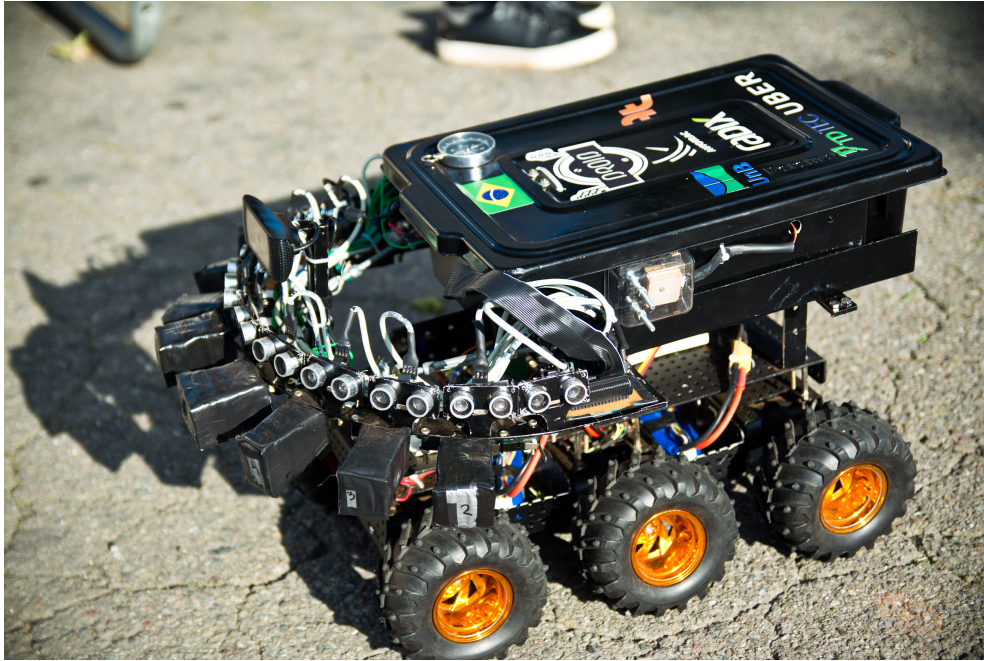


Figura 1.3: Robô de navegação autônoma da equipe DROID

A Figura 1.4 mostra a arquitetura de funcionamento do robô. Ele possui três unidades de processamento, dois Arduinos Mega 2560 e um *RaspberryPi*. Um Arduino é responsável pelo controle de velocidade, atuando sobre os motores e realizando a leitura dos codificadores incrementais. O outro Arduino é responsável pela leitura dos sensores ultrassônicos e dos sensores de toque, bem como pela comunicação com o *RaspberryPi*. O *RaspberryPi* é o verdadeiro cérebro do robô, onde todas as informações são processadas. Ele também realiza as leituras da câmera, da IMU e do receptor de GPS.

A parte abordada neste trabalho compreende o sistema de localização.

1.3 Objetivos do Projeto

O objetivo principal desse trabalho de graduação é desenvolver um sistema de localização para o robô da equipe DROID, visando a participação na categoria de navegação autônoma *RoboMagellan* da competição mundial de robótica *RoboGames*. O projeto é baseado na integração de GPS/IMU/odometria por meio de filtros *bayesianos*, mais especificamente o Filtro de Kalman Estendido (EKF) e o Filtro de Kalman *Unscented* (UKF).

A determinação da localização consiste na determinação da pose do robô (posição e orientação) e de sua velocidade em relação a um sistema de referência tridimensional.

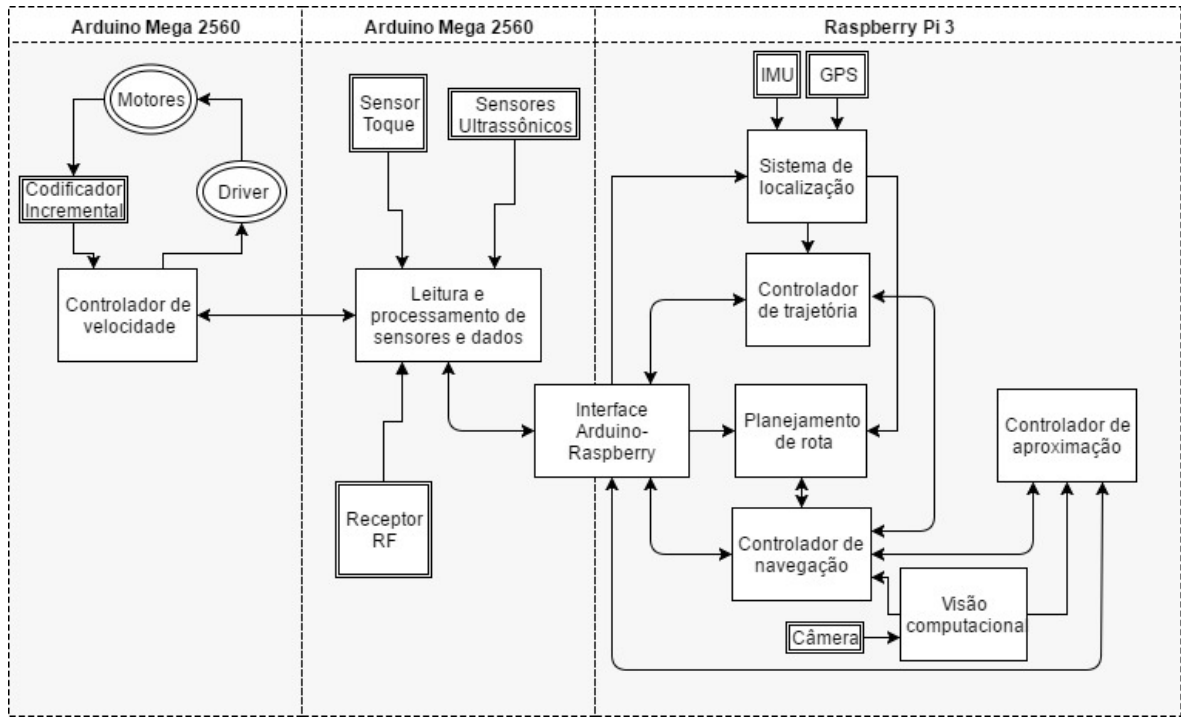


Figura 1.4: Fluxograma da arquitetura do robô da DROID

1.4 Resultados Obtidos

Neste trabalho realizou-se o projeto de sistema de localização para o robô móvel da equipe DROID para navegação em ambientes externos, com a formulação das equações dos filtros utilizados. Foram escritos programas em C++ para coleta de dados da IMU e do GPS e a implementação *offline* do EKF e do UKF no ambiente *MATLAB*. Devido a um problema com a coleta de dados do GPS, foi realizada a estimação da posição e da velocidade do robô apenas por meio de odometria e os filtros foram utilizados na estimação da atitude, validando a abordagem adotada.

1.5 Apresentação do Manuscrito

No Capítulo 2 é apresentada uma visão geral sobre as técnicas de localização de robôs móveis em ambientes externos, bem como a fundamentação matemática e apresentação das técnicas utilizadas no trabalho. O Capítulo 3 apresenta todo o desenvolvimento do projeto, com a descrição detalhada da abordagem adotada, a modelagem matemática do robô e derivação das equações utilizadas nos filtros, a descrição dos sensores utilizados e, por fim, como se deu a implementação do projeto, com a descrição do *software* desenvolvido. No Capítulo 4 são mostrados os resultados obtidos na coleta de dados dos sensores utilizados e a filtragem *offline* dos mesmos por meio do Filtro de Kalman Estendido e do Filtro de Kalman *Unscented*. O Capítulo 5 apresenta as conclusões do trabalho, bem como propostas de trabalhos futuros. Os anexos apresentam a descrição do conteúdo do CD e contém um material complementar sobre filtros de Kalman e quatérnios na representação de atitude.

Capítulo 2

Fundamentos

2.1 Técnicas de Localização

A fim de navegar autonomamente e realizar tarefas úteis, um robô móvel precisa saber sua exata posição e orientação. Dessa forma, a localização, ou seja, a determinação da pose e da velocidade de um objeto que se movimenta em relação a uma referência conhecida, é essencial para promover real autonomia a um robô móvel.

As técnicas de localização são baseadas em dois principais métodos, o relativo e o absoluto [1, 14].

2.1.1 Métodos Relativos

Os métodos relativos, também conhecidos como métodos de localização local, calculam a pose atual do robô em relação a uma pose inicial. Esse cálculo é feito a partir da movimentação do robô, por meio de medições de distância percorrida, velocidade e tempo decorrido desde a última medição. Dessa forma, sabendo a posição atual em que se encontra, o robô se movimenta e efetua uma estimativa de onde deveria estar após a execução de tal movimento, com base em seus sensores internos (ou proprioceptivos) e da informação do mapa do ambiente ou de seu modelo cinemático.

Como a estimativa é baseada em poses anteriores, o erro da solução de posição e orientação dos métodos relativos é integrado com o passar do tempo, não sendo recomendado para movimentos por longas distâncias, além de necessitar de uma pose inicial conhecida. Os métodos relativos, porém, proveem medições constantes, o que significa que uma estimação da pose estará sempre disponível. A Figura 2.1 ilustra o método de localização relativa.

Na robótica, os métodos de localização relativa se baseiam, em geral, na *odometria* e na *navegação inercial* [5].

Odometria

A odometria é um método que determina a localização de um robô móvel através da observação e integração consecutiva do movimento de suas rodas, sendo possível calcular a distância percorrida.

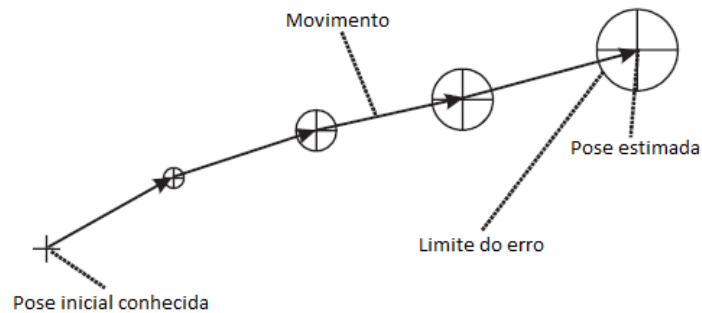


Figura 2.1: Método de localização relativa [1]

O sensor mais usado para isso é o codificador incremental (ou *encoder* em inglês), colocado nas rodas do robô e capaz de contar o número de revoluções dado por cada uma.

A odometria, porém, por ser baseada na integração de informação sobre movimentos incrementais, acarreta em uma grande acumulação de erros. Os erros sistemáticos — devido a características do robô ou do sensor — podem ocorrer devido ao desalinhamento das rodas do robô ou raios das rodas diferentes do nominal. Esses fatores são graves pois são fonte constante de erros aditivos, sendo a fonte principal de erros em solos regulares. Os erros não sistemáticos — característicos da relação do robô com o ambiente — podem ser causados por movimento sobre solos não-uniformes, por movimento sobre obstáculos inesperados no chão e pelo escorregamento ou derrapagem das rodas devido a solos escorregadios, grandes acelerações ou rotações rápidas. Em solos irregulares esses fatores são a fonte predominante de erro.

Navegação Inercial

Sensores inerciais compreendem acelerômetros, que medem força específica, e giroscópios, que medem velocidade angular. Eles são chamados de inerciais pois não precisam de referências externas para realizarem suas medições e, quando em conjunto, são conhecidos como unidade de medição inercial ou IMU (do inglês *Inertial Measurement Unit*) [1].

Um sistema de navegação inercial ou INS (do inglês *Inertial Navigation System*) é formado por uma IMU e por uma unidade de processamento, que integra as saídas dos sensores inerciais para obter posição, velocidade e atitude. A IMU combina três acelerômetros mutuamente ortogonais e três giroscópios alinhados com os acelerômetros, resultando em um sistema de navegação tridimensional completo. Os projetos recentes de INS empregam todos uma arquitetura *strapdown*, de forma que os sensores inerciais são fixos no corpo em estudo.

Os sensores inerciais mais usados atualmente utilizam tecnologia MEMS (do inglês *Micro Electro Mechanical Systems*), pois oferecem a vantagem de terem baixo custo, terem tamanho e peso extremamente reduzidos e alta tolerância a impactos, apesar de apresentarem desempenho mais baixo do que modelos mais caros.

No **acelerômetro**, uma massa de prova é livre para se mover em relação ao encapsulamento da

IMU ao longo do eixo de sensibilidade, sendo restringido por molas, conforme mostrado na Figura 2.2. Quando uma força aceleradora é aplicada ao encapsulamento, ao longo do eixo de sensibilidade, a massa de prova mantém sua velocidade devido à inércia e, assim, o encapsulamento irá se movimentar em relação à massa de prova. A diferença de posição da massa de prova em relação ao encapsulamento é proporcional à força aplicada ao mesmo e, ao medir esse deslocamento, é possível obter uma medida de aceleração. A única exceção é a aceleração devido à força da gravidade, que atua na massa de prova diretamente e aplica a mesma aceleração a todos os componentes da IMU, não havendo movimento relativo entre um e outro. Assim, diz-se que o acelerômetro mede a força específica, ou seja, a aceleração não-gravitacional, e não a aceleração total.

O **giroscópio** pode operar por meio de vários princípios diferentes, sendo os principais tipos os de massa giratória, os ópticos e os vibratórios. Todos os giroscópios MEMS, no entanto, operam por meio do princípio vibratório. Um giroscópio vibratório compreende um elemento que é submetido a um simples movimento harmônico, com o objetivo de detectar a aceleração de Coriolis quando o giroscópio é rotacionado. Essa aceleração provoca um movimento oscilatório perpendicular ao eixo principal de oscilação, cuja amplitude é proporcional à velocidade angular. Um modelo do giroscópio vibratório é mostrado na Figura 2.3.

Os acelerômetros e giroscópio apresentam, no entanto, fontes de erro. Ainda de acordo com [1], esses erros são devidos ao *bias*, um erro constante que independe do valor medido; ao fator de escala, um fator multiplicativo que insere erros maiores quanto maior o valor medido; ao acoplamento cruzado, que surge do desalinhamento dos eixos do sensor em relação aos eixos do sistema de coordenada do corpo; e ao ruído aleatório, que pode ser introduzido por conta de ruídos elétricos, vibrações mecânicas e outros.

Os erros de uma solução de navegação inercial crescem com o tempo à medida que os sucessivos erros do acelerômetro e do giroscópio são somados. Assim, apesar de operar continuamente e oferecer uma solução tridimensional de atitude, posição e velocidade, a acurácia da solução é deteriorada com o passar do tempo.

Apesar de não ser um sensor proprioceptivo, grande parte das IMUs comercializadas vem também com um **magnetômetro**, um sensor que mede a intensidade do campo magnético local. Ele é usado para corrigir a atitude, supondo-se que a fonte principal dos valores medidos é devido

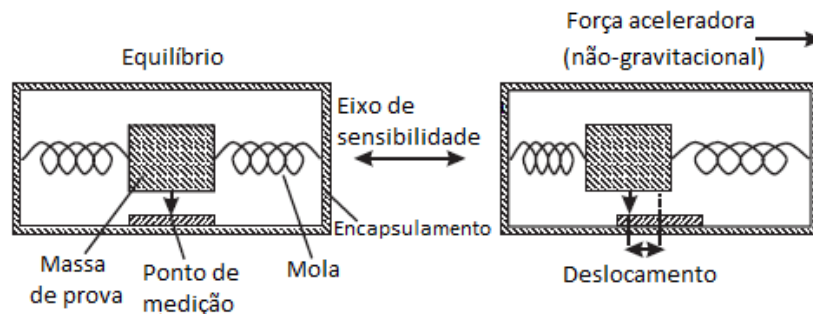


Figura 2.2: Esquema simplificado de um acelerômetro [1]

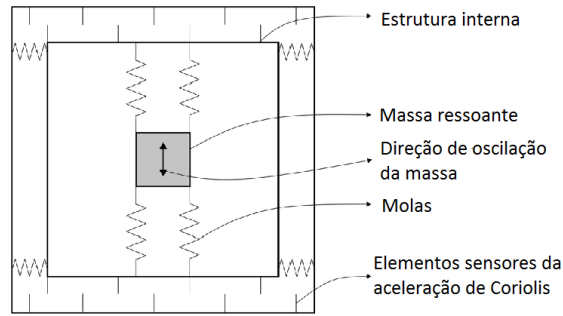


Figura 2.3: Esquema simplificado de um giroscópio vibratório

ao campo magnético terrestre. Essa suposição nem sempre é verdadeira, visto que se pode ter a presença de outros campos magnéticos, como aquele devido aos motores, além de possíveis interações com materiais ferromagnéticos próximos ao sensor.

Em um sistema baseado em coordenadas geográficas, porém, as medições do magnetômetro não podem ser diretamente utilizadas como estimativa de orientação, uma vez que os pólos magnéticos da Terra não coincidem com os pólos geográficos. Para se encontrar o ângulo em relação ao norte geográfico é necessário somar o valor da declinação magnética local ao ângulo em relação ao norte magnético. As informações sobre a intensidade do campo magnético local, bem como a declinação, podem ser encontrados em mapas de campo magnético, como o disponível pela *National Oceanic and Atmospheric Administration* dos Estados Unidos¹. Os valores do campo magnético em Brasília são mostrados na Tabela 2.1.

Elemento Magnético	Valor
Declinação	-21° 30' 47"
Intensidade total ($\ \mathbf{m}^n\ $)	23455.5 nT
Componente X (m_x^n)	19440.8 nT
Componente Y (m_y^n)	-7663.1 nT
Componente Z (m_z^n)	-10653.4 nT

Tabela 2.1: Elementos do campo magnético em Brasília

2.1.2 Métodos Absolutos

Os métodos absolutos, também conhecidos como métodos de *localização global*, são baseados na percepção do ambiente. Eles fornecem a pose do robô diretamente de uma única medição, sem necessitar das poses calculadas anteriormente. Para isso são usados sensores externos (ou exteroceptivos) para detectar marcos ou pontos de referência, que nada mais são do que características no ambiente que o robô pode usar para determinar sua pose.

Um dos métodos mais usados de localização absoluta faz uso do chamado *sistema global de navegação por satélite*.

¹Disponível em <https://www.ngdc.noaa.gov/geomag-web/>

Sistema Global de Navegação por Satélite

O sistema global de navegação por satélite ou GNSS (do inglês *Global Navigation Satellite System*) é o termo usado para descrever os sistemas de navegação que fornecem ao usuário uma solução tridimensional de sua posição, através da transmissão de sinais de rádio de satélites em órbita a receptores na Terra [1].

Os principais sistemas de navegação são o GPS, o GLONASS e Galileo. O GPS (*Global Positioning System*) foi o primeiro sistema GNSS, lançado nos anos 1970 pelos Estados Unidos. Ele consiste em uma constelação de 27 satélites e oferece cobertura global. O GLONASS é o sistema da Rússia, formado por 24 satélites e também oferece cobertura global. Já o Galileo é um sistema GNSS civil da União Europeia, ainda em fase de desenvolvimento, que contará com 27 satélites.

O posicionamento via GNSS é baseado em um processo chamado trilateração, ilustrado na Figura 2.4. Em um espaço bidimensional, caso não se saiba qual a localização de algum ponto, mas se conheça a sua distância a três outros pontos, pode-se calcular sua posição. No espaço tridimensional, como é o caso da localização via GNSS, é necessário conhecer a distância a quatro satélites. O processo de posicionamento se dá da seguinte forma: os satélites GNSS que orbitam a Terra conhecem a efeméride de suas órbitas e o tempo de forma muito precisa, que são ajustados por estações terrestres quando necessário. Os satélites, então, transmitem esses dados e seus sinais de rádio passam pela atmosfera até o receptor. O receptor recebe dados de vários satélites e, para cada satélite, recupera a informação transmitida e, com base no tempo de propagação e velocidade do sinal, consegue computar sua posição.

O posicionamento via GNSS, no entanto, apresenta muitas fontes de erro que podem tirar a acurácia do cálculo da posição. De acordo com [2], essas fontes são a dessincronização dos relógios dos satélites, erros de órbita, a não uniformidade da propagação do sinal na atmosfera terrestre, gerando atrasos ionosféricos e troposféricos e a ocorrência de múltiplos caminhos devido a reflexões e refrações dos sinais, ou ainda, sua obstrução. Alguns destes efeitos de propagação estão retratados na Figura 2.5.

O arranjo geométrico dos satélites em relação ao receptor também afeta a acurácia da posição calculada, conforme mostrado na Figura 2.6. Receptores são idealmente projetados para usar os

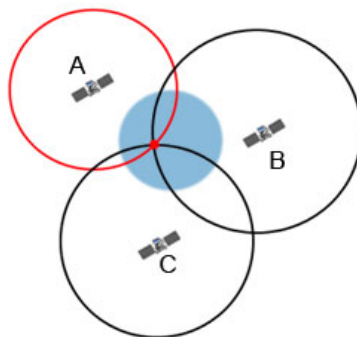


Figura 2.4: Princípio da trilateração

sinais dos satélites disponíveis de forma a minimizar a chamada diluição de precisão (em inglês *dilution of precision*). A diluição de precisão pode ser quantificada pelo receptor durante a determinação de sua posição, sendo possível utilizá-la em algoritmos de localização como uma métrica relativa à qualidade da medida disponível.

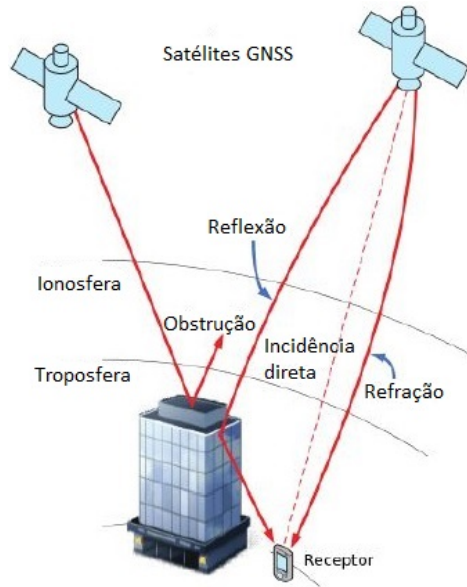


Figura 2.5: Fontes de erro na propagação de sinal do GPS [2]

Além de medidas de posição, receptores GNSS são capazes de fornecer estimativas precisas de velocidade a partir do efeito Doppler. Por serem baseadas em fenômenos distintos, as medições de posição e velocidade fornecidas pela maioria dos receptores GNSS podem ser consideradas descorrelacionadas entre si.

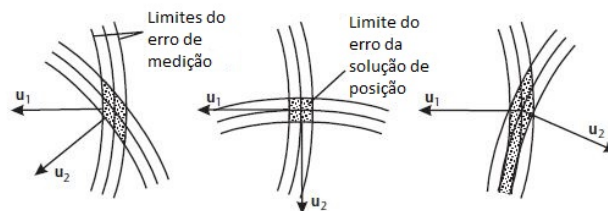


Figura 2.6: Efeito da disposição geométrica dos sinais na acurácia da estimativa de posição[1]

2.1.3 Fusão Sensorial

A análise dos métodos de localização permite concluir que as características dos métodos relativos e absolutos são complementares. O erro na solução de posição dos métodos relativos cresce exponencialmente, porém, fornecem a solução continuamente. Já os métodos absolutos não sofrem de erros crescentes, contudo, geralmente não fornecem soluções contínuas. Dessa forma, torna-se apropriado combinar as duas técnicas em um sistema de navegação integrado, aproveitando os

benefícios de ambos os métodos e o usando o maior número de informação possível.

Assim, algoritmos de fusão sensorial são aqueles que combinam dados relativos e absolutos para melhor estimar a pose do robô. A fusão das informações de diversos sensores é importante, em particular quando nem todos eles medem a mesma coisa. Juntos, os sensores podem prover uma imagem mais completa do ambiente, reduzindo os efeitos dos erros de medição. Como todos os métodos de localização lidam com incertezas, a fusão sensorial é estudada por meio de uma abordagem probabilística.

2.2 Fundamentos Matemáticos

2.2.1 Sistemas de Coordenadas

A definição de sistemas de coordenadas é essencial para qualquer projeto de localização. Neste trabalho, é necessário determinar a posição do robô em relação a referências geográficas. Os sistemas utilizados são mostrados na Figura 2.7 e detalhados abaixo.

Sistema ECEF ou Sistema E

O sistema de coordenadas centrado e fixo na Terra, o ECEF (do inglês *Earth Centred, Earth Fixed*) é referenciado neste trabalho como sistema E. A sua origem está localizada no centro de massa da Terra e é fixo na mesma, isto é, rotaciona junto com a Terra. O eixo Z aponta para o Polo Norte e o eixo X aponta para a interseção do ponto de latitude e longitude 0° e, assim, o plano XY define o plano equatorial.

Para transformar as coordenadas geodésicas de latitude, longitude e altitude para coordenadas ECEF, é necessário o uso de um elipsoide de referência para aproximar o formato da Terra. Um elipsoide de referência pode ser descrito por uma série de parâmetros que definem seu formato: o semi-eixo maior (a), o semi-eixo menor (b), sua excentricidade (e) e seu achatamento (f). Para aplicações globais, a referência geodésica usada para o GPS é o WGS84 (do inglês *World Geodetic System 1984*). Seu elipsoide tem origem coincidente com a origem do sistema ECEF e seus parâmetros são:

- Semi-eixo maior: $a = 6378137 \text{ m}$;
- Semi-eixo menor: $b = 6356752.31424518 \text{ m}$;
- Achatamento: $f = \frac{a-b}{a} = 0.003334 \text{ m}$; e
- excentricidade: $e = \sqrt{f(2-f)} = 0.0818191908426215 \text{ m}$.

Para converter a latitude (λ), longitude (A) e altitude (h) para coordenadas no sistema ECEF, é necessário calcular o raio da curvatura da primeira vertical (N), mostrada na Equação 2.1a. Os cálculos das coordenadas XYZ são dados por

$$N = \frac{a}{\sqrt{1 - e^2 \sin(\lambda)^2}}, \quad (2.1a)$$

$$x_{ecef} = (N + h) \cos(\lambda) \cos(\Lambda), \quad (2.1b)$$

$$y_{ecef} = (N + h) \cos(\lambda) \sin(\Lambda), \quad (2.1c)$$

$$z_{ecef} = (N(1 - e^2) + h) \sin(\lambda). \quad (2.1d)$$

Sistema NED ou Sistema N

De posse das coordenadas no sistema ECEF, pode-se convertê-las para o sistema de navegação, o sistema NED (do inglês *North-East-Down*), referenciado neste trabalho como sistema N. Sua origem pode ser colocada em qualquer ponto da superfície da Terra, geralmente coincidindo com a origem do sistema de coordenadas do robô, em sua posição inicial. O eixo X aponta para o norte geográfico, o eixo Y aponta para o leste e o Z , eixo vertical, aponta para o centro da Terra, de acordo com a regra da mão direita. Assim o plano XY forma um plano tangente à superfície terrestre.

As coordenadas no sistema NED podem ser relacionadas com as coordenadas no sistema ECEF de acordo com a Equação 2.2, em que \mathbf{r}^n é a posição tridimensional que se deseja encontrar no sistema NED, \mathbf{r}^e é a posição correspondente no sistema ECEF e \mathbf{r}_{ref}^e , posição de referência, corresponde ao ponto de origem do sistema NED em coordenadas ECEF. A matriz de transformação M_n^e entre os dois sistemas pode ser definida através da latitude (λ_{ref}) e longitude (Λ_{ref}) do ponto de referência \mathbf{r}_{ref}^e . Ela é dada por

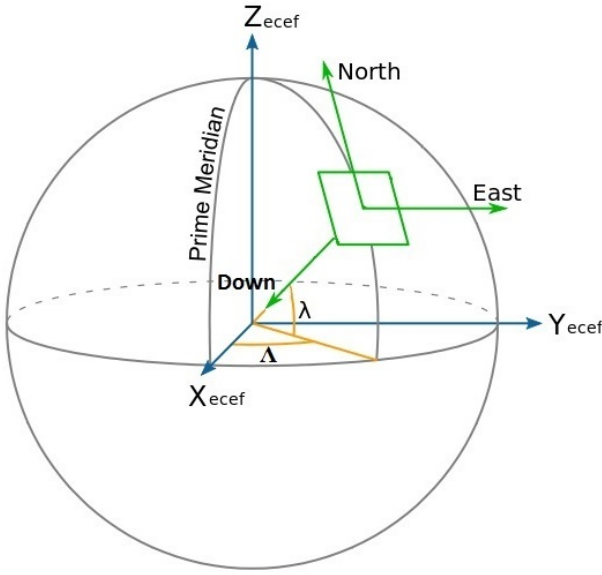
$$\mathbf{r}^n = M_n^e \times (\mathbf{r}^e - \mathbf{r}_{ref}^e), \quad (2.2)$$

em que

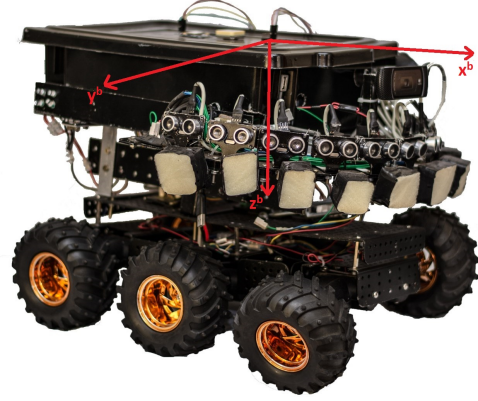
$$M_n^e = \begin{pmatrix} -\sin(\lambda_{ref}) \cos(\Lambda_{ref}) & -\sin(\lambda_{ref}) \sin(\Lambda_{ref}) & \cos(\lambda_{ref}) \\ -\sin(\Lambda_{ref}) & \cos(\Lambda_{ref}) & 0 \\ -\cos(\lambda_{ref}) \cos(\Lambda_{ref}) & -\cos(\lambda_{ref}) \sin(\Lambda_{ref}) & -\sin(\lambda_{ref}) \end{pmatrix}. \quad (2.3)$$

Sistema fixo ao robô ou Sistema B

O sistema B é um sistema de coordenadas fixado no corpo do robô, preferencialmente em seu centro de gravidade. No caso do presente trabalho, a origem deste sistema coincide com a origem do sistema de coordenadas da IMU, e as medidas dos sensores inerciais são tomadas em torno de seus eixos.



(a) Sistemas de coordenadas ECEF e NED



(b) Sistemas de coordenadas fixo ao robô

Figura 2.7: Sistemas de coordenadas usados neste trabalho.

2.2.2 Algoritmo TRIAD

Determinar a atitude do robô é equivalente a determinar a matriz de rotação que descreve a orientação do sistema de referência fixo ao robô (sistema B) em relação ao sistema de navegação (sistema N), ou seja, encontrar \mathbf{R}_n^b . Um método muito conhecido para determinar essa matriz é o algoritmo TRIAD, proposto por Harold D.Black em [15]. Esse é um método determinístico que, dados dois vetores unitários (\mathbf{u} e \mathbf{v}) conhecidos ortonormais expressos nas coordenadas do sistema de referência e do sistema do corpo, encontra a matriz de rotação. Essa técnica, apesar de necessitar de dois vetores não colineares, torna-se vantajosa por ser determinístico em que a atitude é reconstruída a cada instante.

O algoritmo é dado por

$$\mathbf{R}_n^b = \begin{bmatrix} \mathbf{i}^b & \mathbf{j}^b & \mathbf{k}^b \end{bmatrix} \begin{bmatrix} \mathbf{i}^n & \mathbf{j}^n & \mathbf{k}^n \end{bmatrix}^T, \quad (2.4a)$$

$$\mathbf{i} = \mathbf{u}, \quad (2.4b)$$

$$\mathbf{j} = \frac{\mathbf{u} \times \mathbf{v}}{|\mathbf{u} \times \mathbf{v}|}, \quad (2.4c)$$

$$\mathbf{k} = \mathbf{i} \times \mathbf{j}. \quad (2.4d)$$

Neste trabalho, usa-se o vetor de campo magnético local \mathbf{m} e o de aceleração da gravidade \mathbf{g} , de forma que $\mathbf{m}^n = \mathbf{R}_n^b \mathbf{m}^b$ e $\mathbf{g}^n = \mathbf{R}_n^b \mathbf{g}^b$. Os vetores \mathbf{m}^n e \mathbf{g}^n são conhecidos, fixos e constantes

e os vetores \mathbf{m}^b e \mathbf{g}^b são mensuráveis, dados pelas leituras do magnetômetro e do acelerômetro, respectivamente. Como o acelerômetro dá a medida da força específica que age sobre o robô e não só da componente de aceleração da gravidade, esses dados devem ser filtrados, de forma que outras acelerações não sejam levadas em conta. Para isso, utilizou-se a abordagem adotada em [16], em que simplesmente se rejeita aquelas medidas do acelerômetro em que $\|\mathbf{f}^b - \mathbf{g}^b\| > e$. O valor de e foi determinado experimentalmente. No conjunto de Equações 2.4, \mathbf{u} e \mathbf{v} são os vetores unitários correspondentes a \mathbf{g} e \mathbf{m} , respectivamente.

Apesar da atitude ser dada no formato de matriz de rotação, ela pode ser convertida para representação com quatérnio de maneira simples, conforme Equação III.15 do Anexo III.

2.2.3 Filtragem Bayesiana

As incertezas inerentes aos sensores e atuadores de um robô fazem com que a representação de sua pose no mundo através de uma única hipótese seja impossível. Dessa forma, a localização autônoma é um dos principais problemas estudados pela robótica probabilística, em que o robô representa a crença em sua localização por meio de múltiplas hipóteses, expressando explicitamente sua incerteza na estimação de sua pose.

De acordo com [17], a crença reflete o conhecimento interno do robô sobre o estado do ambiente, atribuindo uma probabilidade (ou densidade) para cada possível hipótese de estado em relação ao estado verdadeiro. A crença é representada por meio de distribuições de probabilidade condicional e, por isso, utiliza-se muito o teorema de Bayes, dado por

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}, \quad (2.5)$$

em que x é uma quantidade que se deseja inferir de y , sendo que y representa dados, como, por exemplo, medições feitas por um sensor. A probabilidade $p(x)$, chamada de probabilidade a priori, resume o conhecimento que se tem da variável aleatória X antes de incorporar os dados de y . A probabilidade condicional $p(x|y)$ é chamada de distribuição de probabilidade a posteriori de X , ou seja, a probabilidade de X assumir o valor x , dado que se tem a medição y .

Os algoritmos mais comuns para computar a crença da localização de um robô são, então, baseados na chamada filtragem Bayesiana. Ela é uma técnica recursiva que calcula a crença a partir de dados de medição e do efeito de ações de controle do robô [18]. A densidade de probabilidade $p(X_t = x | X_{t-1} = x', \mathbf{y}_t, \mathbf{u}_t)$ traduz a crença do robô sobre o conjunto de poses que possa estar assumindo no instante t com base em sua crença anterior, medições e comandos de controle realizados.

Dessa forma, estão representadas os dois modelos probabilísticos da dinâmica do robô e seu ambiente: o modelo de movimento do robô, que caracteriza como o estado muda com o tempo com o efeito de uma ação de controle, correspondente à densidade condicional $p(X_t = x | X_{t-1} = x', \mathbf{u}_t)$, em que \mathbf{u}_t é a ação de controle que dá a informação do deslocamento entre os instantes $t - 1$ e t ; e o modelo de observação, que caracteriza a crença no valor medido pelo sensor dado que o robô se encontra em um determinado estado, expressa pela densidade condicional $p(Y_t = y | X_t = x)$. Esses

modelos probabilísticos representam a incerteza inerente à evolução do estado e ao sensoriamento. O Algoritmo 1 mostra as duas etapas básicas da localização baseada em filtragem Bayesiana [17].

Algoritmo 1: ETAPAS DA LOCALIZAÇÃO DA FILTRAGEM BAYESIANA

Entrada: medições y_t , dados de controle u_t , crença da localização anterior $bel(X_{t-1} = x')$

Saída: crença da localização atual $bel(X_t = x)$

```

1 início
2   para toda possível localização  $x$  faça
3     Predição:  $\overline{bel}(X_t = x) \leftarrow \int p(X_t = x | X_{t-1} = x', \mathbf{u}_t) bel(X_{t-1} = x') dx$ 
4     Correção:  $bel(X_t = x) \leftarrow p(Y_t = y | X_t = x) \overline{bel}(X_t = x)$ 
5   fim
6 fim
7 retorna  $bel(X_t = x)$ 

```

A técnica mais estudada e mais famosa para implementar o Filtro de Bayes é o chamado **Filtro de Kalman** (KF), um algoritmo de processamento de dados recursivo que estima o estado de um sistema dinâmico linear ruidoso, proposto por Rudolph Emil Kalman em 1960 [19]. Ele é considerado um filtro gaussiano, em que a crença é representada por uma distribuição normal, caracterizada por dois parâmetros: a média e a covariância. Gaussianas são unimodais, ou seja, apresentam um único ponto de máximo. Isso significa que, no contexto da localização, a saída do filtro de Kalman é a distribuição de possíveis posições do robô, em que a média está localizada em torno da posição real com uma pequena margem de incerteza.

O KF é considerado um estimador de estados ótimo, se o sistema dinâmico puder ser descrito por equações lineares. No mundo real, no entanto, a maioria dos processos é não-linear. Dessa forma, existem extensões do Filtro de Kalman para lidar com a não-linearidade.

A mais famosa dessas extensões é o **Filtro de Kalman Estendido** (EKF) [16, 20, 21, 22], que promove a linearização do modelo do sistema e do modelo de medição por meio da expansão da série de Taylor de primeira ordem. Dessa forma, o EKF herda do Filtro de Kalman a representação básica de crença, porém essa crença é apenas aproximada e não exata. Além disso, a linearização exige a derivação das jacobianas dos modelos, que pode ser um processo moroso.

Outra abordagem para lidar com a não-linearidade é o **Filtro de Kalman Unscented** (UKF), proposto por Julier S. J. et al. [23, 24, 25], desenvolvido por Wan E.A. et al. [26] e sistematizada por Menegaz H. M. T. et al. [27]. O UKF utiliza uma transformação *unscented* para estimar o estado de sistemas não-lineares sem realizar a linearização dos modelos do sistema e de medição. Essa transformação usa um conjunto de amostras (chamado de sigma-pontos) cuidadosamente escolhidas de forma determinística, que parametrizam a média e a covariância da crença. A função do sistema é aplicada a cada amostra, o que resulta em um grupo de pontos transformados. A média e a covariância desse grupo de pontos são a média e a covariância propagada da crença. O UKF trabalha com a premissa de que é mais fácil aproximar uma distribuição gaussiana por meio de um número fixo de parâmetros do que por meio da aproximação de uma função não-linear.

O UKF torna-se uma alternativa atrativa, uma vez que evita o cálculo das jacobianas, fornece

uma performance superior em uma complexidade computacional equivalente ao EKF e, além disso, leva a uma convergência mais rápida em caso de condições iniciais imprecisas [3]. A Figura 2.8 mostra uma comparação entre o UKF e o EKF. As equações dos Filtros de Kalman são descritas no Anexo II.

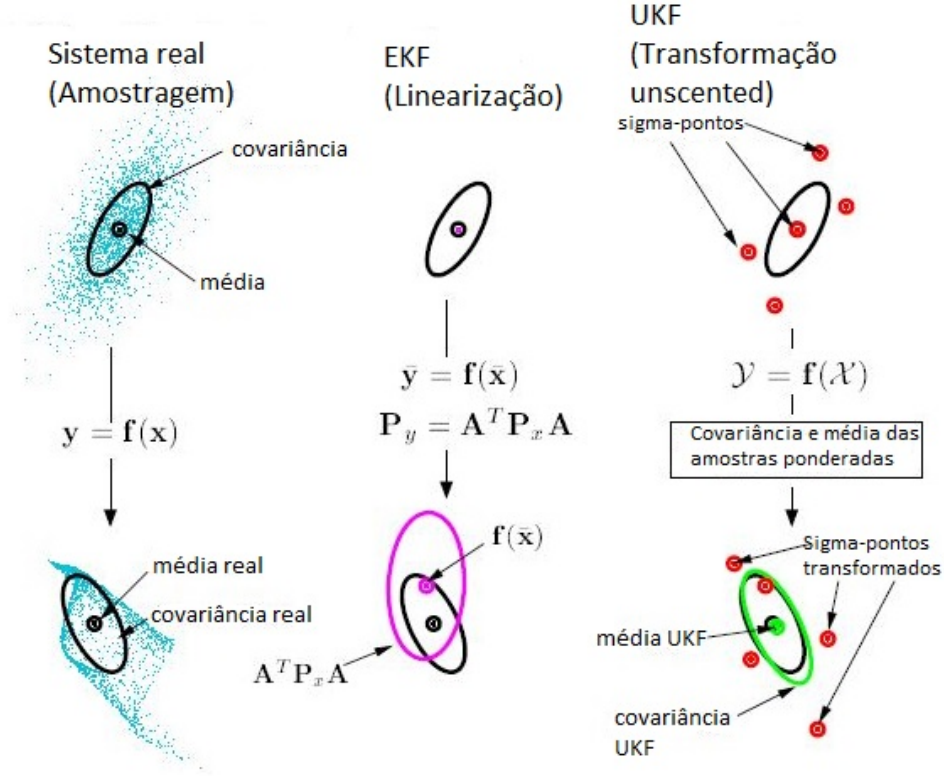


Figura 2.8: Comparação entre a amostragem real, a linearização de primeira ordem do EKF e a transformação *unscented* do UKF [3]

2.2.3.1 Algoritmo USQUE para Estimação de Atitude

A representação da atitude por meio de ângulos de Euler é amplamente usada e de fácil visualização. Porém, apresenta singularidades para determinadas atitudes. Dessa forma, em muitos casos, é preferível utilizar a representação por quatérnios, explicada no Anexo III

A representação de atitude por meio de quatérnios só é possível por meio de quatérnios unitários. Essa restrição às vezes é violada, tanto pela linearização proposta pelo EKF bem como pelo cálculo da média dos sigma-pontos propagados, por meio da soma dos quatérnios, proposto pela abordagem tradicional do UKF.

Para contornar esse problema, Crassidis et al. propôs em [28] um novo algoritmo para estimação de atitude com quatérnios usando a transformação *unscented*, batizado de USQUE (do inglês *UnScented Quaternion Estimator*).

Essa abordagem faz uso do fato de que a covariância 4×4 do quatérnio, por ter posto 3, pode ser projetada em uma matriz 3×3 sem perda de informação. Dessa forma, em vez de se

utilizar diretamente o quatérnio, utiliza-se o erro multiplicativo do quatérnio, através de um vetor de erro de três componentes, eliminando as restrições. A representação escolhida para o vetor de erro foi a de parâmetros generalizados de Rodrigues, em que a singularidade pode ser colocada em qualquer lugar entre $\pm 180^\circ$ e $\pm 360^\circ$. Como é utilizado um vetor de apenas três componentes, essa singularidade nunca é encontrada na prática. As correções são realizadas utilizando o produto de quatérnio, garantido a restrição da unicidade.

As equações que transformam o quatérnio em um vetor de erro de quatérnio e deste para a representação de parâmetros de Rodrigues e vice-versa são mostradas no Anexo II.

Capítulo 3

Desenvolvimento

3.1 Abordagem Adotada

Este trabalho utiliza o filtro de Kalman Estendido e o filtro de Kalman *Unscented* para realizar a fusão dos dados fornecidos pelos sensores e determinar a localização tridimensional do robô no sistema N .

O receptor de GPS fornece as estimativas da posição do robô no sistema E (\mathbf{r}^e) e da velocidade do robô no sistema N (\mathbf{v}^n). A IMU — dotada de giroscópio, acelerômetro e magnetômetro triaxiais — fornece as medições de velocidade angular (ω_{nb}^b), força específica (\mathbf{f}^b) e campo magnético (\mathbf{m}^b) no sistema B . Já a odometria fornece as estimativas de velocidades angulares das rodas ($\dot{\varphi}$). As Figuras 3.1 e 3.2 mostram os diagramas do EKF e do UKF, respectivamente.

Torna-se importante destacar que, por mais que o projeto seja de localização tridimensional, usa-se a odometria bidimensional para estimar a posição nas coordenadas XY , deixando a estimação no eixo Z apenas para a correção com o GPS. A odometria foi utilizada para estimar a posição no lugar da integração das medições do acelerômetro e do giroscópio, que forneceria uma estimativa tridimensional, pois, em momentos em que o sinal do GPS está indisponível ou com baixa qualidade, a odometria é um método de estimação mais confiável do que a fornecida pela IMU, cuja solução diverge muito rapidamente. Além disso, a própria estrutura do robô, que não é muito robusta, favorece a vibração do mesmo, aumentando ainda mais a deriva da solução.

Os vetores de estado, de entrada e de medição são dados por

$$\mathbf{x} = \begin{bmatrix} \mathbf{q}_n^b \\ \mathbf{r}^n \\ \mathbf{v}^n \end{bmatrix}_{10 \times 1}, \quad \mathbf{u} = \begin{bmatrix} \omega_{nb}^b \\ \dot{\varphi} \end{bmatrix}_{5 \times 1}, \quad \mathbf{y} = \begin{bmatrix} \mathbf{q}_n^b \\ \mathbf{r}^n \\ \mathbf{v}^n \end{bmatrix}_{10 \times 1}. \quad (3.1)$$

em que vetor de estado \mathbf{x} é composto pelo quatérnio de atitude (\mathbf{q}_n^b), o vetor de posição (\mathbf{v}_n) e o vetor de velocidade do robô (\mathbf{r}_n), expressos no sistema N , totalizando 10 variáveis de estado. O vetor de entrada \mathbf{u} é composto pelas medições do giroscópio (ω_{nb}^b) e da odometria ($\dot{\varphi}$). Já o vetor de medição \mathbf{y} é dado pela estimativa do quatérnio de atitude calculado pelo algoritmo TRIAD e

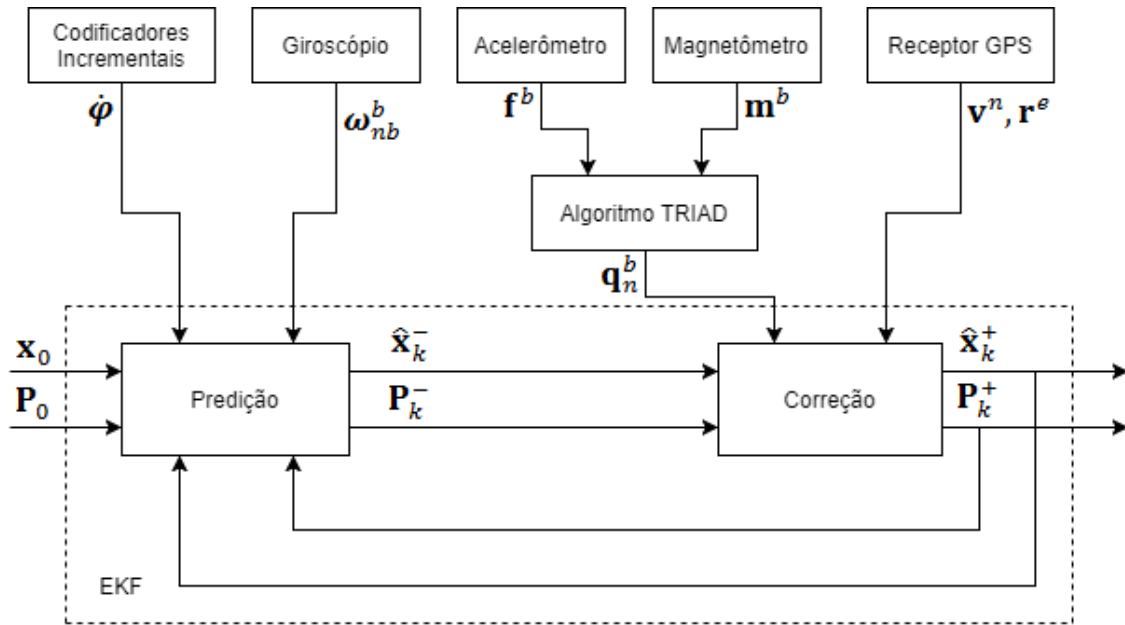


Figura 3.1: Diagrama de blocos do EKF para localização 3D

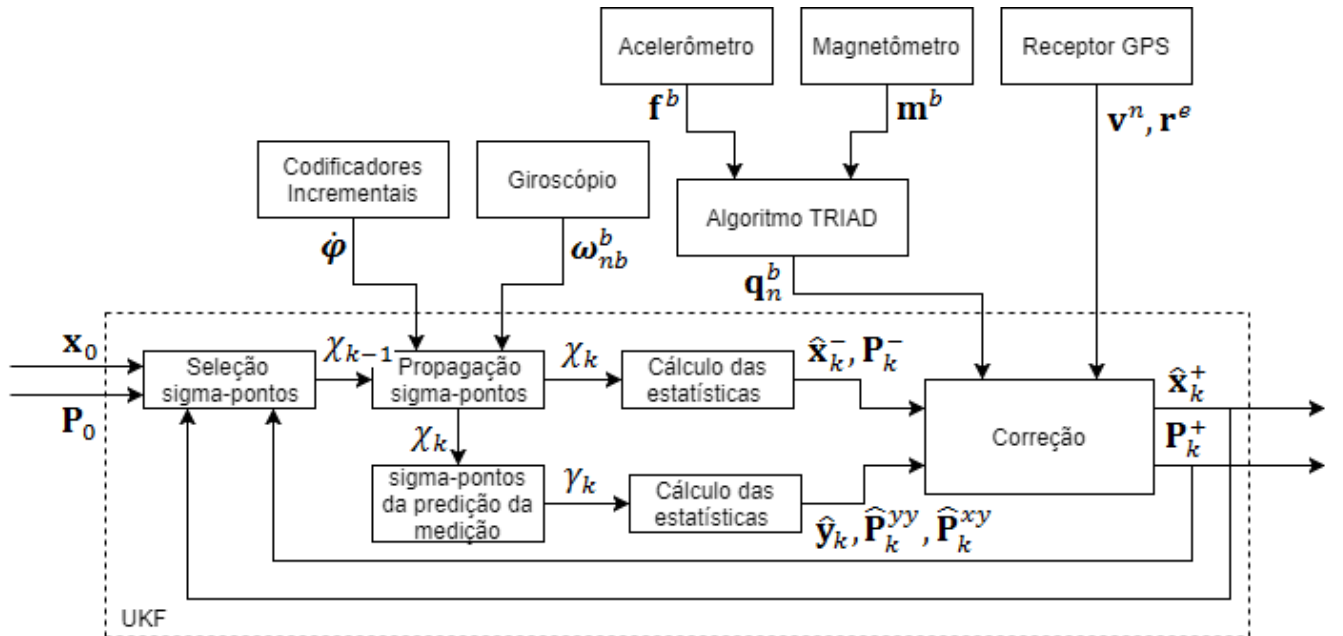


Figura 3.2: Diagrama de blocos do UKF para localização 3D

pelas estimativas de posição e velocidade fornecidas pelo GPS, transformadas para o sistema N.

3.2 Modelo Cinemático do Robô

O primeiro passo para desenvolver o projeto de localização é determinar o modelo cinemático do robô, ou seja, o modelo de seu movimento. De acordo com a técnica apresentada em [29], o modelo cinemático é dado sob a forma direta

$$\dot{\xi} = J(p, \gamma, \psi) \times \dot{p}, \quad (3.2)$$

em que J é a matriz jacobiana do modelo cinemático; p é o vetor de variáveis que interferem na movimentação do robô, tomadas no sistema de coordenadas local do robô $X_B \times Y_B$; e γ representa o conjunto de parâmetros geométricos do robô. A pose do robô, $\xi = (r_x, r_y, \psi)^T$, é dada no sistema de coordenadas global $X_N \times Y_N$.

Para um robô diferencial, $p = (\varphi_d, \varphi_e)^T$, representando os ângulos dos eixos de tração das rodas direita e esquerda, respectivamente. Dessa forma, as velocidades rotacionais das rodas de tração, $\dot{\varphi}_d$ e $\dot{\varphi}_e$, compõe o vetor \dot{p} , de forma que, da Equação 3.2, obtêm-se as velocidades de tração e rotação do robô, v_x , v_y e ω_z , no sistema de coordenadas global. Os parâmetros geométricos do robô, $\gamma = (r, B)^T$, representam o raio das rodas e a distância entre elas. Assim, o modelo cinemático assume a forma

$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = J(p, \gamma, \psi) \times \begin{bmatrix} \dot{\varphi}_d \\ \dot{\varphi}_e \end{bmatrix}. \quad (3.3)$$

A Figura 3.3 mostra os parâmetros do modelo cinemático.

Com base na Figura 3.3, pode-se encontrar a matriz $J(p, \gamma, \psi)$. Sendo CC o centro de curva do robô e R_c o raio dessa curva, pode-se dizer que a velocidade do robô v é dada por

$$v = \omega_z \times R_c. \quad (3.4)$$

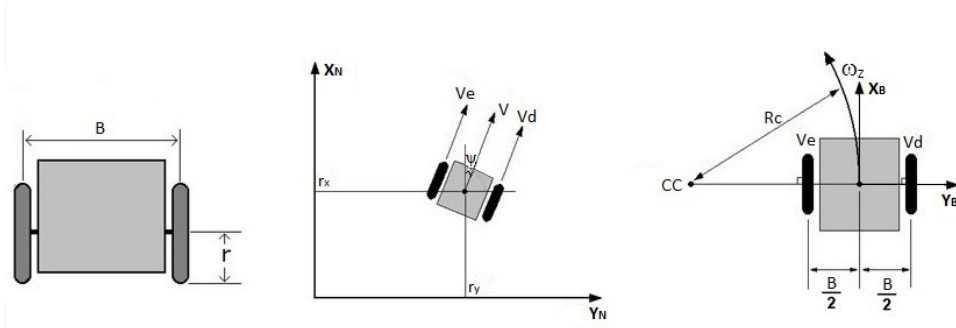


Figura 3.3: Parâmetros que descrevem a pose e a cinemática do robô a tração diferencial

Já as velocidades lineares das rodas da esquerda (v_e) e da direita (v_d) podem ser dadas por

$$v_e = \omega_z \times (R_c - \frac{B}{2}), \quad (3.5)$$

$$v_d = \omega_z \times (R_c + \frac{B}{2}). \quad (3.6)$$

A partir das Equações 3.4 a 3.6 e sabendo que $v_d = \dot{\varphi}_d r$ e $v_e = \dot{\varphi}_e r$, tem-se que

$$v = (\dot{\varphi}_e + \dot{\varphi}_d) \times \frac{r}{2}, \quad (3.7)$$

$$\omega_z = (\dot{\varphi}_d - \dot{\varphi}_e) \times \frac{r}{B}. \quad (3.8)$$

A velocidade v pode, agora, ser decomposta em suas coordenadas v_x e v_y . De acordo com a Figura 3.3, $v_x = v \times \cos(\psi)$ e $v_y = v \times \sin(\psi)$. Substituindo v na Equação 3.7, pode-se definir a matriz J como

$$J(p, \gamma, \psi) = \begin{bmatrix} \frac{r}{2} \cos(\psi) & \frac{r}{2} \cos(\psi) \\ \frac{r}{2} \sin(\psi) & \frac{r}{2} \sin(\psi) \\ \frac{r}{B} & -\frac{r}{B} \end{bmatrix}. \quad (3.9)$$

Tem-se, assim, o modelo cinemático do robô.

3.3 Modelo do Processo

As equações diferenciais que regem a dinâmica do processo são dadas por

$$\dot{\mathbf{q}}_n^b = e - \frac{1}{2} \mathbf{q}_n^b \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}_{nb}^b \end{bmatrix}, \quad (3.10a)$$

$$\dot{\mathbf{r}}^n = \mathbf{v}^n, \quad (3.10b)$$

$$\dot{\xi} = J(p, \gamma, \psi) \times \dot{p}. \quad (3.10c)$$

em que a primeira equação representa a propagação de atitude por quatérnios, dadas as velocidades angulares do robô; a segunda representa o modelo cinemático do robô, discutido na seção anterior; e a terceira representa a integração da velocidade para se obter a posição.

A discretização das equações diferenciais [16], que dão o modelo não-linear do processo, são dadas por

$$\mathbf{q}_k = e^{-\frac{1}{2} \mathbf{W} \Delta t} \mathbf{q}_{k-1}, \quad (3.11a)$$

$$\mathbf{r}_k = \mathbf{r}_{k-1} + \mathbf{v}_{k-1}\Delta t, \quad (3.11b)$$

$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \frac{r}{2}\cos(\psi_{k-1}) & \frac{r}{2}\cos(\psi_{k-1}) \\ \frac{r}{2}\sin(\psi_{k-1}) & \frac{r}{2}\sin(\psi_{k-1}) \\ \frac{r}{B} & -\frac{r}{B} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_d \\ \dot{\varphi}_e \end{bmatrix}, \quad (3.11c)$$

$$v_{z,k} = v_{z,k-1}, \quad (3.11d)$$

em que a matriz \mathbf{W} , mostrada na Equação III.18, é a matriz anti-simétrica das velocidades angulares.

Vale lembrar que a posição e a velocidade no eixo Z são estimadas apenas pela correção do GPS e, por isso, a velocidade no instante atual é igual à do momento anterior, conforme a Equação 3.11d.

3.4 Modelo de Medição

O modelo de medição fornece diretamente a estimação do estado. As leituras fornecidas pelo acelerômetro e pelo magnetômetro são processadas pelo algoritmo TRIAD e a matriz de rotação resultante é transformada no quatérnio por meio do conjunto de Equações III.15. Já o receptor de GPS fornece as velocidades diretamente no sistema N e as posições, fornecidas no sistema E, podem ser convertidas para o sistema N por meio da Equação 2.2. Assim, o modelo de medição é dado por

$$\mathbf{y} = \mathbf{I}_{10 \times 10} \begin{bmatrix} \mathbf{q}_n^b \\ \mathbf{r}^n \\ \mathbf{v}^n \end{bmatrix}. \quad (3.12)$$

3.5 Equações para EKF e UKF

O modelo do sistema em estudo, dado por

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k, \quad \mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k), \quad (3.13a)$$

$$\mathbf{y}_k = h(\mathbf{x}_k) + \mathbf{v}_k, \quad \mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k), \quad (3.13b)$$

representa os modelos de processo e de medição discutidos nas seções anteriores. A função \mathbf{f} é dado pelo conjunto de Equações 3.11 e a função \mathbf{h} é dada pela identidade mostrada na Equação 3.12.

Conforme descrito no Anexo II, o EKF exige a linearização do modelo não-linear descrito, utilizando a expansão da série de Taylor de primeira ordem. As jacobianas do modelo de processo e do modelo de medição foram calculadas e mostradas nas Equações 3.14a e 3.15, respectivamente. A linearização da função das velocidades v_x e v_y , mostrada na Equação 3.11c exigiu a transformação do ângulo de guinada ψ em quatérnio, conforme a Equação III.20, fazendo os ângulos de arfagem e rolagem iguais a zero.

$$\mathbf{F}_k = \frac{\partial f(\mathbf{x}_{k-1}, \mathbf{u}_k)}{\partial \mathbf{x}_{k-1}} = \begin{bmatrix} e^{-\frac{1}{2}\mathbf{W}\Delta t}\mathbf{q}_{k-1} & \vdots & \mathbf{0}_{4 \times 3} & \vdots & \mathbf{0}_{4 \times 3} \\ \dots & & \dots & & \dots \\ \mathbf{0}_{3 \times 4} & \vdots & \mathbf{I}_{3 \times 3} & \vdots & \mathbf{I}_{3 \times 3}\delta t \\ \dots & & \dots & & \dots \\ \frac{\partial v}{\partial \mathbf{q}_n^b} & \vdots & \mathbf{0}_{3 \times 3} & \vdots & \mathbf{0}_{2 \times 3} \\ \dots & & \dots & & \dots \\ \mathbf{0}_{1 \times 4} & \vdots & \mathbf{0}_{1 \times 3} & \vdots & 0 \ 0 \ 1 \end{bmatrix} \quad (3.14a)$$

$$\mathbf{v} = \begin{bmatrix} \frac{r}{2}\cos(\psi_{k-1}) & \frac{r}{2}\cos(\psi_{k-1}) \\ \frac{r}{2}\sin(\psi_{k-1}) & \frac{r}{2}\sin(\psi_{k-1}) \end{bmatrix} \begin{bmatrix} \dot{\varphi}_d \\ \dot{\varphi}_e \end{bmatrix} \quad (3.14b)$$

$$\mathbf{H}_k = \frac{\partial(\mathbf{x}_k)}{\partial \mathbf{x}_k} = \mathbf{I}_{10 \times 10} \quad (3.15)$$

De posse do modelo do processo, do modelo de medição e da linearização do sistema é possível aplicar o Filtro de Kalman Estendido e o Filtro de Kalman *Unscented* de acordo com as etapas delineadas nos Algoritmos 2 e 3. A estimação da atitude no UKF foi obtida utilizando o algoritmo USQUE, discutido na Seção 2.2.3.1. As equações referentes a esse método foram omitidas do Algoritmo 3 para facilitar a visualização, porém, são descritas no Anexo II. Nos algoritmos, os subscritos q, r, v representam as matrizes referentes à atitude, posição e velocidade, respectivamente.

3.6 Descrição dos Componentes

3.6.1 Codificador Incremental

Para realizar a odometria, foram utilizados codificadores incrementais acoplados aos motores do robô. Como o robô é de tração diferencial, apenas um codificador em cada lado é necessário. Assim, os motores do meio do lado esquerdo e do lado direito foram substituídos por motores com codificadores da empresa *pololu*¹.

Esses codificadores são baseados em um sensor de *Hall effect* de dois canais (A e B) é usado para detectar a rotação de um disco magnético acoplado ao eixo do motor. Eles fornecem uma resolução de 48 pulsos por revolução do eixo do motor quando as duas bordas dos dois canais são utilizadas. As saídas dos canais A e B são ondas quadradas que estão defasadas em aproximadamente 90 °. A frequência das transições diz qual a velocidade angular do motor e a ordem das transições dos dois canais diz qual a direção.

¹Produto disponível em <https://www.pololu.com/product/3217>

Algoritmo 2: FUSÃO SENSORIAL GPS/IMU/ODOMETRIA UTILIZANDO O EKF

Entrada: $\mathbf{x}_0, \mathbf{P}_0, \mathbf{Q}, \mathbf{R}$

Saída: $\hat{\mathbf{x}}_k^+, \mathbf{P}_k^+$

1 início

2 **para** *cada dado processado faça*

3 **se** *há dados da IMU então*

4 **Predição da atitude:**

5 $\hat{\mathbf{q}}_k^- \leftarrow f_q(\hat{\mathbf{q}}_{k-1}^+, \mathbf{u}_k)$

6 $\mathbf{P}_{q,k}^- \leftarrow \mathbf{F}_{q,k} \mathbf{P}_{q,k-1}^+ \mathbf{F}_{q,k}^T + \mathbf{Q}_{q,k}$

7

8 **Correção da atitude:**

9 $\mathbf{K}_{q,k} \leftarrow \mathbf{P}_{q,k}^- \mathbf{H}_{q,k}^T (\mathbf{H}_{q,k} \mathbf{P}_{q,k}^- \mathbf{H}_{q,k}^T + \mathbf{R}_{q,k})^{-1}$

10 $\hat{\mathbf{q}}_k^+ \leftarrow \hat{\mathbf{q}}_k^- + \mathbf{K}_{q,k} (\mathbf{y}_k^{[1:4]} - h_q(\hat{\mathbf{q}}_k^-))$

11 $\mathbf{P}_{q,k}^+ \leftarrow (\mathbf{I} - \mathbf{K}_{q,k} \mathbf{H}_{q,k}) \mathbf{P}_{q,k}^- (\mathbf{I} - \mathbf{K}_{q,k} \mathbf{H}_{q,k})^T + \mathbf{K}_{q,k} \mathbf{R}_{q,k} \mathbf{K}_{q,k}^T$

12 **fim**

13 **se** *há dados da Odometria então*

14 **Predição da posição e velocidade:**

15 $\hat{\mathbf{r}}_k^- \leftarrow f_r(\hat{\mathbf{r}}_{k-1}^+, \mathbf{u}_k)$

16 $\hat{\mathbf{v}}_k^- \leftarrow f_v(\hat{\mathbf{v}}_{k-1}^+, \mathbf{u}_k)$

17 $\mathbf{P}_{r,v,k}^- \leftarrow \mathbf{F}_{r,v,k} \mathbf{P}_{r,v,k-1}^+ \mathbf{F}_{r,v,k}^T + \mathbf{Q}_{r,v,k}$

18 **fim**

19 **se** *há dados do GPS então*

20 **Correção da posição e velocidade:**

21 $\mathbf{K}_{r,v,k} \leftarrow \mathbf{P}_{r,v,k}^- \mathbf{H}_{r,v,k}^T (\mathbf{H}_{r,v,k} \mathbf{P}_{r,v,k}^- \mathbf{H}_{r,v,k}^T + \mathbf{R}_{r,v,k})^{-1}$

22 $\hat{\mathbf{r}}_k^+ \leftarrow \hat{\mathbf{r}}_k^- + \mathbf{K}_{r,v,k} (\mathbf{y}_k^{[5:7]} - h_r(\hat{\mathbf{r}}_k^-))$

23 $\hat{\mathbf{v}}_k^+ \leftarrow \hat{\mathbf{v}}_k^- + \mathbf{K}_{r,v,k} (\mathbf{y}_k^{[8:10]} - h_v(\hat{\mathbf{v}}_k^-))$

24 $\mathbf{P}_{r,v,k}^+ \leftarrow (\mathbf{I} - \mathbf{K}_{r,v,k} \mathbf{H}_{r,v,k}) \mathbf{P}_{r,v,k}^- (\mathbf{I} - \mathbf{K}_{r,v,k} \mathbf{H}_{r,v,k})^T + \mathbf{K}_{r,v,k} \mathbf{R}_{r,v,k} \mathbf{K}_{r,v,k}^T$

25 **fim**

26 **fim**

27 **fim**

Algoritmo 3: FUSÃO SENSORIAL GPS/IMU/ODOMETRIA UTILIZANDO O UKF E O US-QUE

Entrada: $\mathbf{x}_0, \mathbf{P}_0, \mathbf{Q}, \mathbf{R}$

Saída: $\hat{\mathbf{x}}_k^+, \mathbf{P}_k^+$

1 início

2 **para** *cada dado processado faça*

3 **se** *há dados da IMU então*

4 **Predição da atitude:**

5 Cálculo dos sigma-pontos do vetor de erro ($\chi_{k-1}^{\delta d}$)

6 Cálculo dos sigma-pontos do quatérnio de erro ($\chi_{k-1}^{\delta q}$)

7 Cálculo dos sigma-pontos do quatérnio nominal (χ_{k-1}^q)

8 Propagação dos sigma-pontos do quatérnio nominal (χ_k^q)

9 Cálculo dos sigma-pontos propagados do quatérnio de erro ($\chi_k^{\delta q}$)

10 Cálculo dos sigma-pontos propagados do vetor de erro ($\chi_k^{\delta d}$)

11 Cálculo da média e da covariância do vetor de erro ($\hat{\mathbf{x}}_k^{\delta d,-}$ e \mathbf{P}_k^-)

12 Cálculo dos sigma-pontos da predição da medição (γ_k)

13 Cálculo da média e das covariâncias da predição da medição ($\hat{\mathbf{y}}_{q,k}$, $\hat{\mathbf{P}}_{q,k}^{yy}$ e $\hat{\mathbf{P}}_{q,k}^{xy}$)

14

15 **Correção da atitude:**

16 $\mathbf{K}_{q,k} \leftarrow \hat{\mathbf{P}}_{q,k}^{xy} (\hat{\mathbf{P}}_{q,k}^{yy})^{-1}$

17 $\hat{\mathbf{x}}_k^{\delta d,+} \leftarrow \hat{\mathbf{x}}_k^{\delta d,-} + \mathbf{K}_{q,k} (\mathbf{y}_k^{[1:4]} - \hat{\mathbf{y}}_{q,k})$

18 $\mathbf{P}_{q,k}^+ \leftarrow \mathbf{P}_{q,k}^- (\mathbf{K}_{q,k} \hat{\mathbf{P}}_{q,k}^{yy} \mathbf{K}_{q,k}^T)$

19 Cálculo da estimativa corrigida do quatérnio de erro ($\hat{\mathbf{x}}_k^{\delta q,+}$)

20 Cálculo da estimativa corrigida do quatérnio unitário ($\hat{\mathbf{x}}_k^{q,+}$)

21 **fim**

22 **se** *se há dados da Odometria então*

23 **Predição da posição e velocidade:**

24 Cálculo dos sigma-pontos (χ_{k-1})

25 Propagação dos sigma-pontos (χ_k)

26 Cálculo da média e da covariância ($\hat{\mathbf{x}}_{r,v,k}$ e \mathbf{P}_k^-)

27 Cálculo dos sigma-pontos da predição da medição (γ_k)

28 Cálculo da média e covariância da predição da medição ($\hat{\mathbf{y}}_{r,v,k}$, $\hat{\mathbf{P}}_{r,v,k}^{yy}$ e $\hat{\mathbf{P}}_{r,v,k}^{xy}$)

29 **fim**

30 **se** *se há dados do GPS então*

31 **Correção da posição e velocidade:**

32 $\mathbf{K}_{r,v,k} \leftarrow \hat{\mathbf{P}}_{r,v,k}^{xy} (\hat{\mathbf{P}}_{r,v,k}^{yy})^{-1}$

33 $\hat{\mathbf{r}}_k^+ \leftarrow \hat{\mathbf{r}}_k^- + \mathbf{K}_{r,v,k} (\mathbf{y}_k^{[5:7]} - \hat{\mathbf{y}}_k^{[5:7]})$

34 $\hat{\mathbf{v}}_k^+ \leftarrow \hat{\mathbf{v}}_k^- + \mathbf{K}_{r,v,k} (\mathbf{y}_k^{[8:10]} - \hat{\mathbf{y}}_k^{[8:10]})$

35 $\mathbf{P}_{r,v,k}^+ \leftarrow \mathbf{P}_{r,v,k}^- (\mathbf{K}_{r,v,k} \hat{\mathbf{P}}_{r,v,k}^{yy} \mathbf{K}_{r,v,k}^T)$

36 **fim**

37 **fim**

38 **fim**

3.6.2 Receptor de GPS

Como um dos objetivos do projeto é manter um custo baixo, o primeiro receptor de GPS utilizado foi o *EM-406A*, da *GloboSat Technology*, que a equipe já possuía. Ele utiliza o *chipset SiRF StarIII*, com *cold start* de 42 s e *hot start* de 1 s, possui uma acurácia de 5 metros e fornece os dados a cada 1 s, aproximadamente, por meio de comunicação serial com taxa de 4800 bps. As mensagens de saída seguem o protocolo *NMEA 0183*.

Esse receptor, no entanto, era pouco preciso e sua antena já havia sido reparada algumas vezes. Isso fez com que a solução de posição fornecida pelo receptor fosse degradada, não atendendo às demandas do problema. Assim, optou-se por trocar o receptor.

O novo receptor de GPS escolhido foi o *NEO-6M* da *u-blox* [30], mostrado na Figura 3.4. As Tabelas 3.1 e 3.2 mostram as características do receptor. Apesar de o módulo da *u-blox* possuir interfaces UART, USB e SPI, o encapsulamento comprado suporta apenas a interface UART, por meio dos pinos *Rx* e *Tx*. A taxa padrão de comunicação é de 9600 bps, podendo ser configurada.

Parâmetro	Especificação
Tempo até a primeira estimativa válida	<i>Cold Start</i> : 27 s <i>Warm Start</i> : 27 s <i>Hot Start</i> : 1 s <i>Aided Start</i> : 1 s
Acurácia de posição horizontal	2.5 m
Acurácia de velocidade	0.1 m
Máxima taxa de aquisição de dados	5 Hz

Tabela 3.1: Performance do GPS *NEO-6M*

A análise da Tabela 3.1 permite perceber a superioridade do receptor escolhido em relação ao antigo, principalmente em termos de acurácia e em termos de taxa de aquisição de dados. Outra grande vantagem do novo receptor é a disponibilidade de outros protocolos. O protocolo padrão NMEA não fornece a velocidade tridimensional, apenas uma projeção da mesma sobre o solo (o chamado *speed over ground*). Já o protocolo UBX, proprietário da *u-blox*, fornece inúmeros

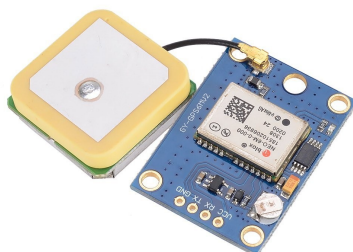


Figura 3.4: GPS NEO-6M da *u-blox*

Protocolo	Tipo
NMEA	Entrada/Saída, ASCII, 0183
UBX	Entrada/Saída, binário, proprietário <i>u-blox</i>
RTCM	Entrada

Tabela 3.2: Protocolos disponíveis para o GPS *NEO-6M*.

dados sobre a navegação, como posição em termos de latitude, longitude e altitude, a posição e a velocidade no sistema ECEF, a velocidade no sistema NED, além de fornecer a acurácia das leituras. A especificação dos protocolos disponíveis para o *NEO-6M* pode ser encontrado em [31].

Dessa forma, optou-se por utilizar o protocolo UBX, que fornece diretamente a posição (\mathbf{r}^e) e a velocidade (\mathbf{v}^n).

3.6.3 IMU

A IMU empregada neste projeto foi a MPU-9250 da *InvenSense* [32], mostrada na Figura 3.5. A MPU-9250 consiste de dois *chips* integrados em um único pacote. O primeiro é a MPU-6500 formado por um acelerômetro e um giroscópio triaxiais. O segundo é o magnetômetro de 3-eixos AK8963, da *Asahi Kasei Microdevices Corporation*. Dessa forma, a MPU-9250 é um dispositivo de rastreamento de movimento de 9 eixos.

A Tabela 3.3 mostra as características da IMU. Para se obter um rastreamento preciso tanto para movimentos rápidos quanto para movimentos lentos, a faixa dinâmica dos sensores é programável.

A descrição do mapa de registradores da MPU-9250 pode ser encontrada em [33].

3.7 Implementação

A implementação do projeto consistiu em três etapas. A primeira delas foi o desenvolvimento de programas em C++ para realizar a coleta dos dados provenientes da IMU e do GPS. A segunda foi a implementação dos algoritmos do Filtro de Kalman Estendido e do Filtro de Kalman *Unscented* no *MATLAB* para análise *offline* dos dados coletados. A terceira foi a implementação em C++ de um sistema de localização em tempo real utilizando apenas a odometria. Os dados da odometria, ou seja, as velocidades angulares das rodas do robô, foram fornecidas pelo Arduino.

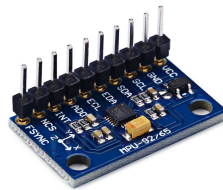


Figura 3.5: IMU MPU-9250 da *InvenSense*

Característica	Especificação
Protocolo de comunicação	I^2C
Resolução	Giroscópio: 16 bits Acelerômetro: 16 bits Magnetômetro: 14 ou 16 bits
Faixa dinâmica	Giroscópio: $\pm 250, \pm 500, \pm 1000, \pm 2000$ ($^{\circ}/s$) Acelerômetro: $\pm 2, \pm 4, \pm 6, \pm 8, \pm 16$ (g) Magnetômetro: $\pm 4800 \mu T$
Sensitividade	Giroscópio: 131, 65.5, 32.8, 16.4 ($LSB/(^{\circ}/s)$) Acelerômetro: 16.384, 8.192, 4.096, 2.048 (LSB/g) Magnetômetro: 0.6, 015 ($\mu T/LSB$)

Tabela 3.3: Características da MPU-9250

Os programas em C++ foram escritos no *RaspberryPi* embarcado no robô, com o uso da ferramenta computacional ROS (do inglês *Robot Operating System*). O ROS é uma plataforma utilizada para simplificar a tarefa de criação de *softwares* para robôs e consiste em uma coleção de estruturas de *software* (bibliotecas, ferramentas e *drivers*) que permitem a transmissão de mensagens, gerenciamento de pacotes, abstração de *hardware* e outras facilidades. O uso do ROS foi essencial para a integração das diversas tarefas realizadas pelo robô.

3.7.1 Coleta de Dados

Para armazenar os dados provenientes dos sensores, foi utilizada a ferramenta *rosvbag*. Uma *bag* é um formato de arquivo do ROS onde são salvos dados com o tempo de aquisição de cada um. É uma ferramenta muito útil para armazenar dados, tanto para uso *online*, na qual as informações salvas podem ser reproduzidas, quanto para uso *offline*, para guardar dados que serão futuramente utilizados.

3.7.1.1 IMU

A IMU e o *RaspberryPi* são interfaceados por meio do barramento I^2C . Para acessar as portas *sda* e *scl* que compõe o barramento, foi utilizada a *wiringPi*, uma biblioteca que realiza o interfaceamento dos pinos GPIO do *RaspberryPi*¹, desenvolvida por Gordon Henderson. Essa biblioteca possui funções para abrir o dispositivo, ler e escrever em seus registradores.

Para cada um dos sensores presentes na IMU foi escrito um programa diferente, seguindo o fluxograma mostrado na Figura 3.7(b).

Para que os sensores funcionem da forma desejada, é necessário configurá-los, por meio da escrita em registradores específicos de configuração. As faixas dinâmicas do acelerômetro e do giroscópio foram configuradas para 16g e 2000($^{\circ}/s$), respectivamente, para permitir uma melhor

¹<http://wiringpi.com/>

detecção de movimentos rápidos. Já o magnetômetro foi configurado para modo de operação contínua, com resolução de 16 bits.

Os laços principais dos programas consistem na leitura dos registradores de saída de cada sensor, com uma taxa de 100 Hz, e o armazenamento dos valores medidos nos eixos XYZ em *bags*, até que o programa seja finalizado.

Antes de armazenar os dados de saída, é necessário tratar os dados brutos lidos diretamente dos sensores. Para isso, deve-se multiplicar o dado bruto pelo fator de sensibilidade configurado, de acordo com a Tabela 3.3.

Além disso, como dito na Seção 2.1.1, as medições dos sensores da IMU são corrompidos por erros, sendo que dois deles podem ser calculados: o *bias* e o fator de escala. Neste trabalho, realizou-se uma calibração prévia dos sensores, em que esses parâmetros foram estimados anteriormente e depois incorporados às medições brutas, de acordo com

$$\mathbf{a} = \frac{\tilde{\mathbf{a}} - \mathbf{b}_a}{s_a}, \quad (3.16)$$

em que a é o valor real da grandeza medida pelo sensor, \tilde{a} é a saída bruta do sensor, já corrigida pelo fator de sensibilidade, b_a é o *bias* e s_a é o fator de escala.

Os fatores de escala e *biases* podem mudar com o passar do tempo então, idealmente, esses parâmetros deveriam fazer parte do estado para serem estimados pelo filtro durante a execução do mesmo. Neste trabalho, no entanto, por praticidade, realizou-se apenas a calibração prévia. A estimação *online* desses parâmetros pode ser um trabalho futuro.

Calibração do Acelerômetro

A calibração do acelerômetro seguiu o procedimento proposto em [16]. Esse procedimento é baseado na aquisição de medidas do sensor quando imóvel, porém, em diversas atitudes, sem haver necessidade de alinhamentos precisos entre as configurações diferentes. Quando parado, o acelerômetro mede apenas a componente da aceleração da gravidade, \mathbf{g}^b . Assim, tem-se

$$\|\mathbf{g}^b\| = \|\mathbf{f}^b\| = \sqrt{(f_x^b)^2 + (f_y^b)^2 + (f_z^b)^2}, \quad (3.17)$$

em que \mathbf{f}^b é a medida do acelerômetro. Substituindo a Equação 3.16 na Equação 3.17, tem-se

$$\|\mathbf{g}^b\| = \sqrt{\frac{\tilde{f}_x^b - b_{fx}}{s_{fx}}^2 + \frac{\tilde{f}_y^b - b_{fy}}{s_{fy}}^2 + \frac{\tilde{f}_z^b - b_{fz}}{s_{fz}}^2}. \quad (3.18)$$

Como \mathbf{g}^b é conhecido, $\mathbf{g}^b = [0 \ 0 \ 9.78]^T$, o problema de se encontrar os parâmetros \mathbf{b}^f e \mathbf{s}^f pode ser resolvido de maneira iterativa, por meio da minimização do erro entre o valor real $\|\mathbf{g}^b\|$ e o valor estimado com os parâmetros $\|\mathbf{f}^b\|$. A técnica utilizada para encontrar os parâmetros que minimizam o erro foi a de Gauss-Newton.

Assim, esse procedimento foi primeiro implementado em *MATLAB*, com dados coletados do acelerômetro, para validar o processo. A Figura 3.6 mostra um exemplo do resultado da estimação

dos parâmetros. Pode-se perceber que os valores convergem após 5 interações.

Depois, foi implementado em C++ no *RaspberryPi* embarcado no robô. O programa feito coleta medidas de 7 atitudes diferentes do robô, calcula os parâmetros pelo método discutido acima e os armazena em uma *rosvbag*, que é posteriormente lido pelo programa de coleta de dados.

Calibração do Giroscópio e do Magnetômetro

Como o giroscópio mede zero quando em repouso independente de sua atitude, o método discutido acima não funciona. Assim, a proposta foi fazer um filtro passa-alta para remover a componente DC do sinal, ou seja, seu *bias*. Esse filtro foi implementado em C++ no programa de coleta de dados.

Como a intensidade do campo magnético local é conhecida, pode-se, supostamente, estimar os parâmetros do magnetômetro pelo mesmo método do acelerômetro. Porém, não foi possível alcançar resultados satisfatórios, uma vez que a estimação dos parâmetros divergia. Dessa forma, optou-se por um método mais simples que apenas corrige o *bias*. Com a ajuda de uma bússola, o robô foi alinhado com o norte magnético e foram feitas leituras durante 30 segundos. Depois foi feita uma média dessas leituras e, como a intensidade do campo magnético é conhecida em cada eixo, de acordo com a 2.1, pôde-se estimar um *bias*.

3.7.1.2 GPS

O GPS e o *RaspberryPi* conversam entre si por meio da interface UART, ou seja, de seus pinos *Rx* e *Tx*. Tentou-se, inicialmente, usar a biblioteca *WiringPi*, como na IMU. Porém, testes iniciais mostraram que os dados estavam sendo transmitido para o *RaspberryPi* com um grande atraso, provavelmente por algum problema relacionado ao *buffer*. Optou-se, então por usar a biblioteca *termios.h*, uma API Unix para lidar com comunicação serial.

O programa para coleta de dados do GPS segue o fluxograma mostrado na Figura 3.7(a).

Após configurar a comunicação serial, definindo o nome da porta e a taxa de transmissão de 9600 bps, é necessário configurar qual o protocolo e quais mensagens serão lidas. Como o protocolo padrão é o NMEA, primeiro a leitura dessas mensagens é desativada e depois habilita-se a leitura das mensagens do protocolo UBX. Foram escolhidas a mensagem *NAV-POSECEF*, que fornece a posição no sistema ECEF, a mensagem *NAV-VELNED*, que fornece a velocidade no sistema NED, além da mensagem *NAV-STATUS*, que fornece o *status* de navegação do GPS, ou seja, se existem estimativas e se essas estimativas são válidas. A taxa de leitura de dados foi configurada para 5 Hz.

A função de leitura das mensagens é detalhada na Figura 3.8, em que *C* é o contador de bytes lidos e *TIPO* é o tipo da mensagem lida. Nessa função, um byte é lido de cada vez e vai sendo armazenado em uma estrutura *union*. O tipo de mensagem é identificado e, quando a mensagem chega ao seu fim, a função retorna esse tipo. Já no laço principal, os dados armazenados na *union* são salvos em *bags*, de acordo com o tipo da mensagem lida.

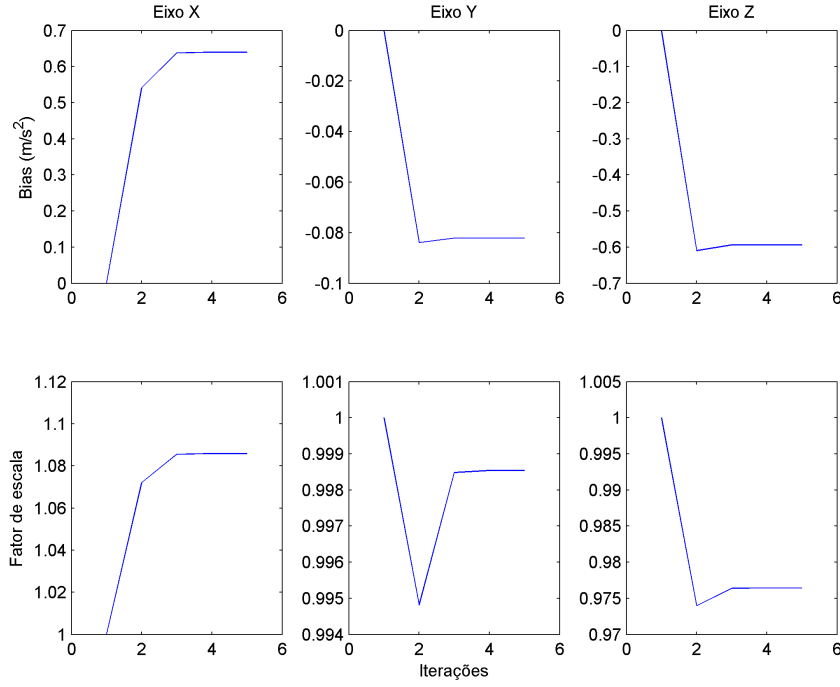


Figura 3.6: Evolução dos parâmetros estimados para o acelerômetro

3.7.2 Implementação dos Filtros de Kalman

Para análise *offline* dos dados coletados, o EKF e o UKF foram implementados em *MATLAB*, conforme as etapas descritas nos Algoritmos 2 e 3. Os programas foram escritos sem o auxílio de nenhuma *toolbox* ou outra ferramenta.

3.7.3 Implementação da Odometria

A fim de realizar testes no mundo real e integrar este trabalho com os trabalhos de planejamento de rota [12] e de controle de movimento [13], foi implementado um primeiro sistema de localização em tempo real baseado apenas na odometria. A implementação foi feita em *C++* no *RaspberryPi*, utilizando a biblioteca matemática *Eigen*.

O programa recebe as mensagens do tópico do ROS de coleta de dados do GPS e faz uma média dos valores de posição lidos nos primeiros 30 segundos de execução, para determinar sua posição inicial. Depois disso, o programa recebe as mensagens do tópico de coleta de dados da odometria e, por meio da implementação da Equação 3.11c, é capaz de fornecer a posição e o ângulo de guinada do robô, dados por

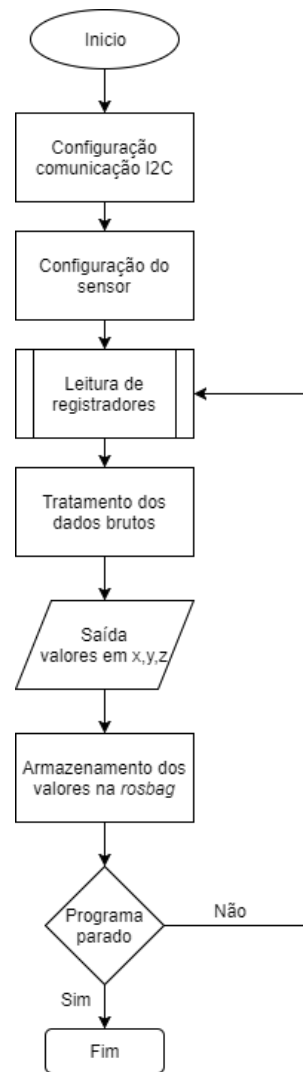
$$\psi_k = \psi_{k-1} + \omega_{z,k} \Delta t, \quad (3.19a)$$

$$r_{x,k} = r_{x,k-1} + v_{x,k} \Delta t, \quad (3.19b)$$

$$r_{y,k} = r_{y,k-1} + v_{y,k} \Delta t. \quad (3.19c)$$



(a) Fluxograma da coleta de dados do GPS



(b) Fluxograma da coleta de dados da IMU

Figura 3.7: Programas de coleta de dados

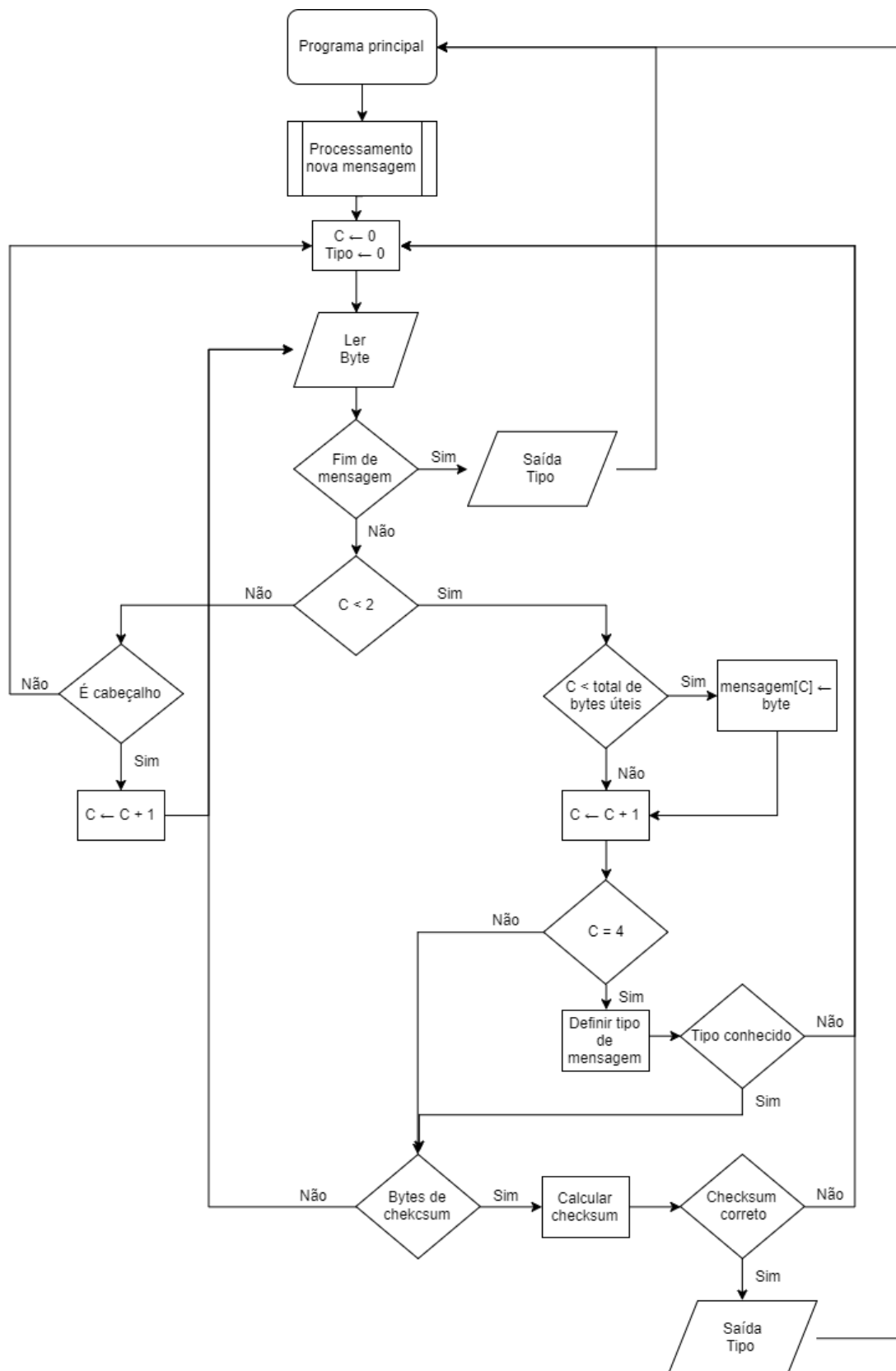


Figura 3.8: Fluxograma da subrotina de processamento das mensagens do GPS

Capítulo 4

Resultados

4.1 Testes de Coleta de Dados

Para realizar a análise dos filtros, foram realizados três testes de coleta de dados. Os testes foram feitos utilizando o próprio robô, dirigido por controle remoto. Dessa forma, foi possível adquirir dados de todos os sensores, que foram armazenados em *bags*, conforme descrito na Seção 3.6.1. Em todas as trajetórias propostas, o robô foi posicionado alinhado com o norte geográfico, com a ajuda de um aplicativo de bússola que já desconta a declinação magnética local.

Como não é possível ter acesso à rota real percorrida pelo robô, os testes foram feitos em caminhos fechados, ou seja, o ponto de chegada do robô é o mesmo do de saída. Assim, pode-se analisar a distância entre esses pontos após a filtragem dos dados como uma medida da qualidade dos algoritmos empregados. As rotas aproximadas feitas pelo robô são mostradas na Figura 4.1, em que os caminhos foram desenhados no *Google Earth* manualmente. As Trajetórias 1, 2 e 3 têm, aproximadamente, 48, 78 e 120 metros.

4.1.1 Atraso do GPS

Após a coleta de dados dos caminhos mostrados na Figura 4.1, foi realizada a análise desses dados. A primeira coisa que se pôde perceber foi atraso das estimativas fornecidas pelo GPS. Acredita-se que, mesmo mudando a biblioteca que lida com a comunicação serial do *RaspberryPi*, as mensagens provenientes do GPS não eram repassadas na velocidade correta, devido a algum atraso de *buffer*. A Figura 4.2 mostra um exemplo de alguns dos pontos entregues pelo programa de coleta de dados do GPS durante o percurso da Trajetória 3, incluindo o último ponto fornecido. Pode-se perceber que os pontos vão apenas até a metade da trajetória, o que significa que o restante dos dados não havia sido entregue ao programa até a finalização do programa.

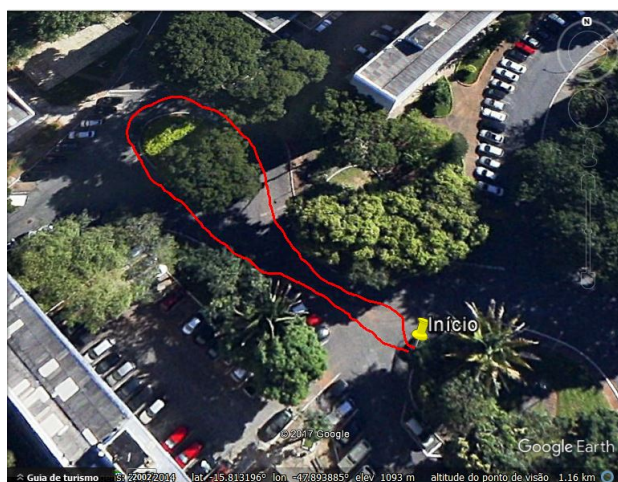
Diante disso, os dados do GPS tornam-se inutilizáveis, visto que representam apenas a metade do caminho. Assim, decidiu-se por implementar apenas o filtro de atitude, deixando a estimação da velocidade e da posição apenas com a odometria.



(a) Trajetória 1 - volta em torno de uma quadra de aproximadamente 8×16 m



(b) Trajetória 2 - volta em torno de uma quadra de aproximadamente 14×25 m



(c) Trajetória 3 - volta em torno de uma rotatória com quebra-molas no caminho

Figura 4.1: Caminhos fechados realizados pelo robô



Figura 4.2: Pontos fornecidos pelo programa de coleta de dados do GPS durante o percurso da Trajetória 3

4.1.2 Taxas de Amostragem

Um parâmetro muito importante em sistemas discretos é a taxa de amostragem, uma vez que ela pode alterar completamente o resultado, caso apresente valor incorreto. Assim, para implementar os filtros, verificou-se se as taxas de amostragem estavam de acordo com aquelas definidas pelo ROS, nos programas de coleta de dados. Os resultados são mostrados nas Figuras 4.3 e 4.4.

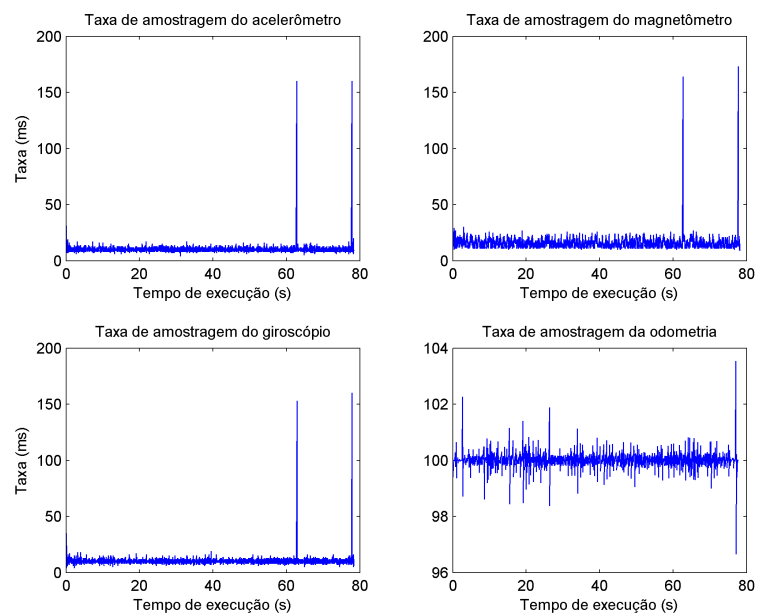
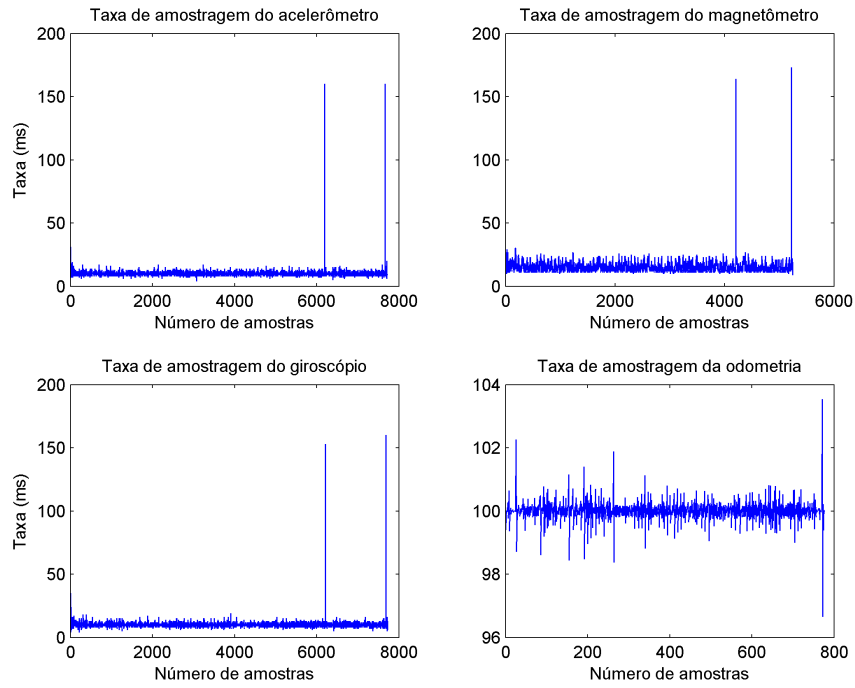
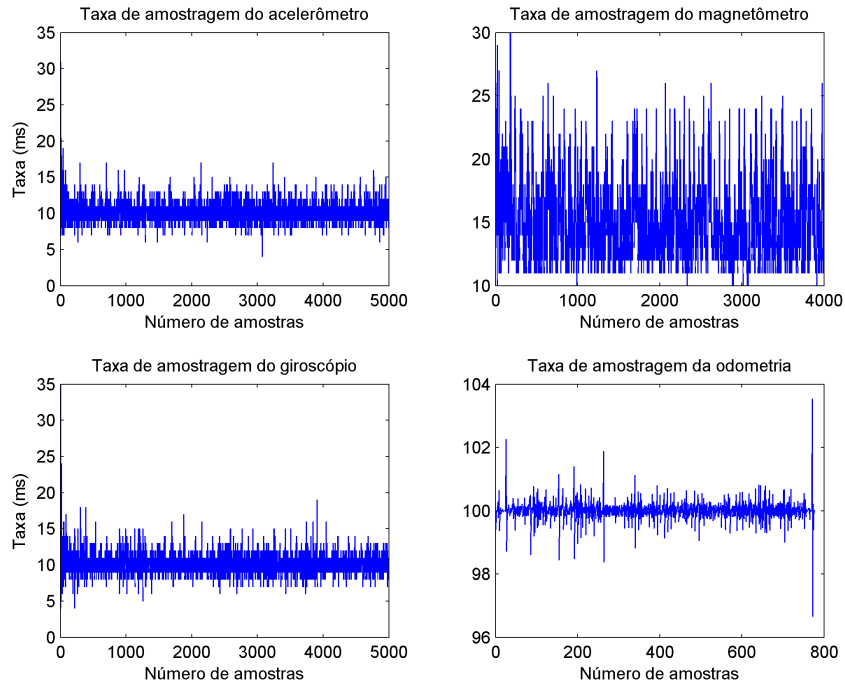


Figura 4.3: Taxa de amostragem dos sensores durante o percurso da Trajetória 1 pelo tempo de execução



(a) Taxa de amostragem vs número de amostras



(b) Detalhe da taxa de amostragem, sem os picos

Figura 4.4: Taxa de amostragem dos sensores durante o percurso da Trajetória 1 pelo número de amostras

As taxas de amostragem, apesar de variarem muito, ficam em torno de um único valor, definido no programa de coleta de dados. A única exceção é o magnetômetro, que apresentou uma taxa maior do que a definida, em torno de 15 ms, além de apresentar um número total de amostras menor que o do acelerômetro e giroscópio. Também pode-se perceber que os três sensores que compõem a IMU tiveram dois grandes picos de atraso ao mesmo tempo, devendo ter sido causado por algum problema de comunicação com a IMU como um todo, afetando as medições de todos os sensores. A odometria é a que apresenta uma maior ocorrência de picos. Isso se deve ao fato de suas leituras serem provenientes do Arduino, que também envia outras informações ao *RaspberryPi*, o que gera atrasos.

4.1.3 Performance do Modelo de Processo e do Modelo de Medição

Após a análise da taxa de amostragem, foi analisada a performance do sistema de localização quando é utilizada apenas a predição da atitude, ou seja, utilizando apenas o modelo do processo de propagação de atitude com as velocidades angulares fornecidas pelo giroscópio; e quando é utilizada apenas a correção da atitude, ou seja, utilizando apenas o modelo de medição, com a atitude fornecida pelo algoritmo TRIAD. Os resultados são mostrados nas Figuras 4.5 a 4.7.

A estimação da posição é dada apenas pela odometria e, assim, influenciada diretamente pela estimação da orientação.

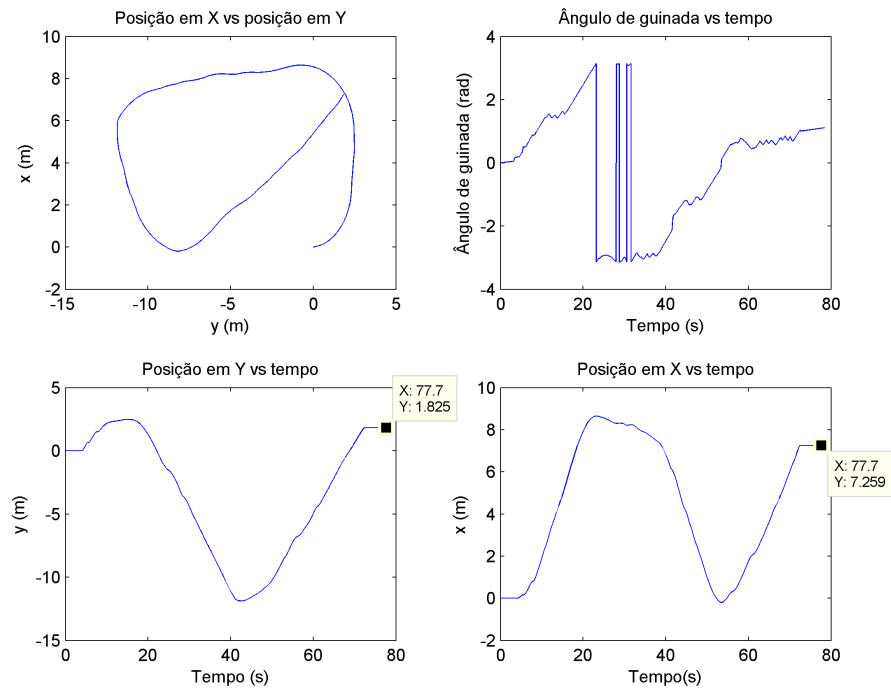
A análise das Figuras 4.5 a 4.7 permite perceber a grande deriva a que os métodos relativos estão sujeitos, em especial na Figura 4.7(a), em que a presença de dois quebra-molas no caminho fez com que a posição estimada se afastasse muito da real. Pode-se perceber também que o algoritmo TRIAD faz um bom trabalho na estimação absoluta da orientação.

4.1.4 Fusão Sensorial

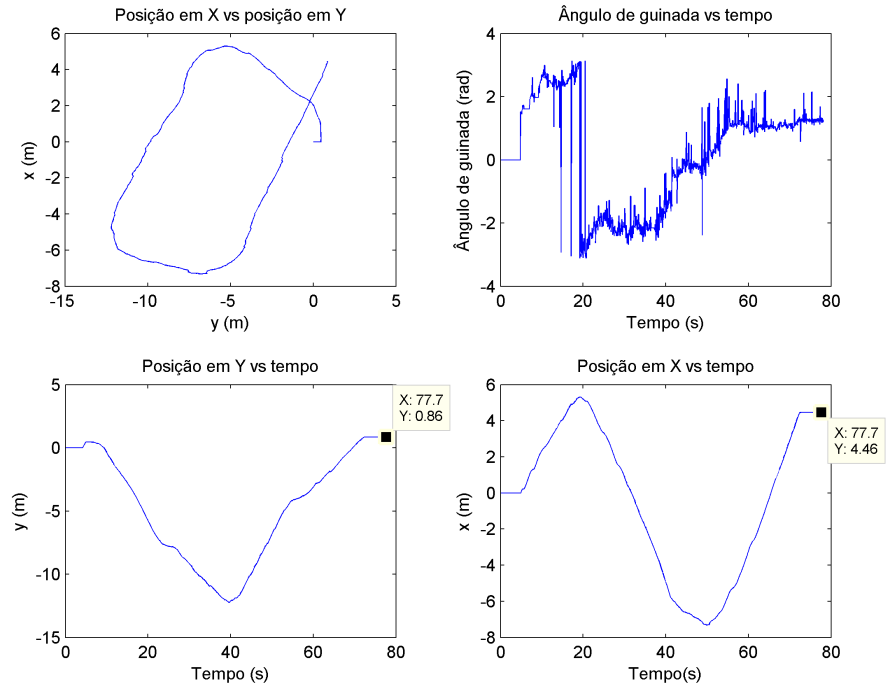
O Filtro de Kalman Estendido e o Filtro de Kalman *Unscented* foram, então aplicados, de forma a realizar a fusão dos dados e obter uma melhor estimação do que aquelas obtidas apenas com a predição ou apenas com a medição. Os parâmetros dos filtros, isto é, as matrizes de covariância do ruído de processo \mathbf{Q} e de medição \mathbf{R} , bem como a inicialização da matriz de covariância da estimação \mathbf{P}_0 , foram determinados empiricamente, sendo mantidos constantes entre os dois filtros. Como pode-se observar nas Figuras 4.5 a 4.7, pode-se confiar mais no modelo de predição que no modelo de processo e, assim, a matriz \mathbf{Q} possui um valor maior que a matriz \mathbf{R} .

Os resultados são mostrados nas Figuras 4.8 a 4.10. Pode-se observar que o Filtro de Kalman *Unscented* teve uma performance levemente melhor que a do Filtro de Kalman Estendido, visto que erro entre a posição final e inicial do robô foi menor. O tempo de execução do Filtro de Kalman *Unscented*, porém, foi um pouco superior, visto que o algoritmo USQUE exige inúmeras propagações dos sigma-pontos.

Foi analisado também que, quando aumenta-se a confiança no modelo de medição e diminui-se a confiança no modelo de processo, as respostas dos dois filtros vão ficando cada vez mais

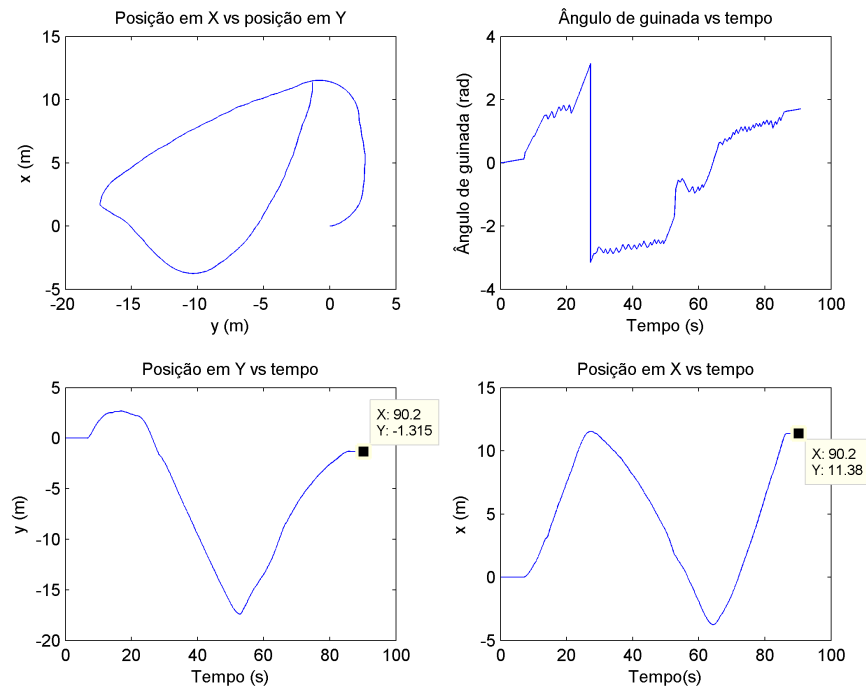


(a) Pose estimada apenas pelo modelo do processo com medidas do giroscópio

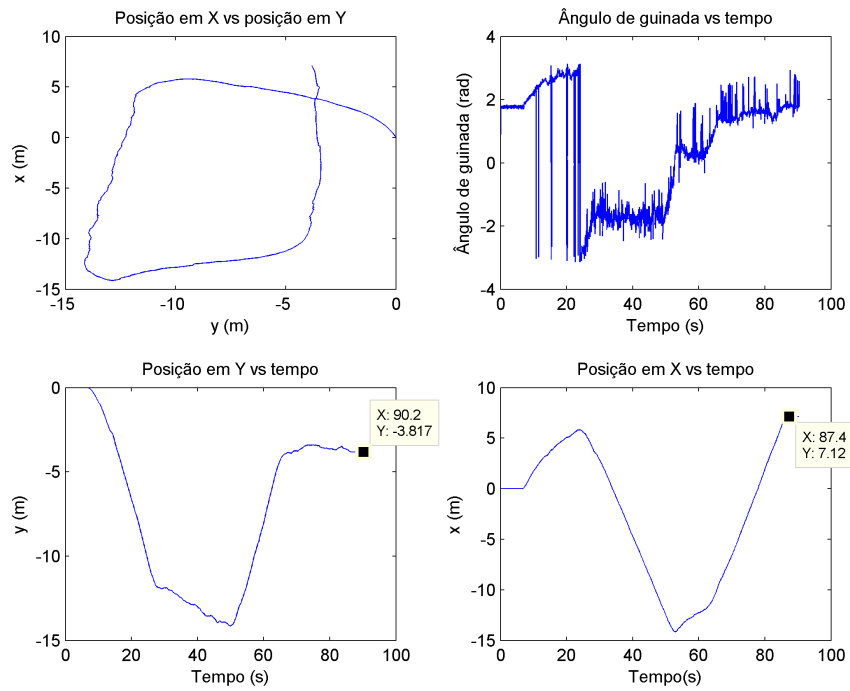


(b) Pose estimada apenas pelo modelo de medição com o algoritmo TRIAD

Figura 4.5: Estimação da pose do robô na Trajetória 1

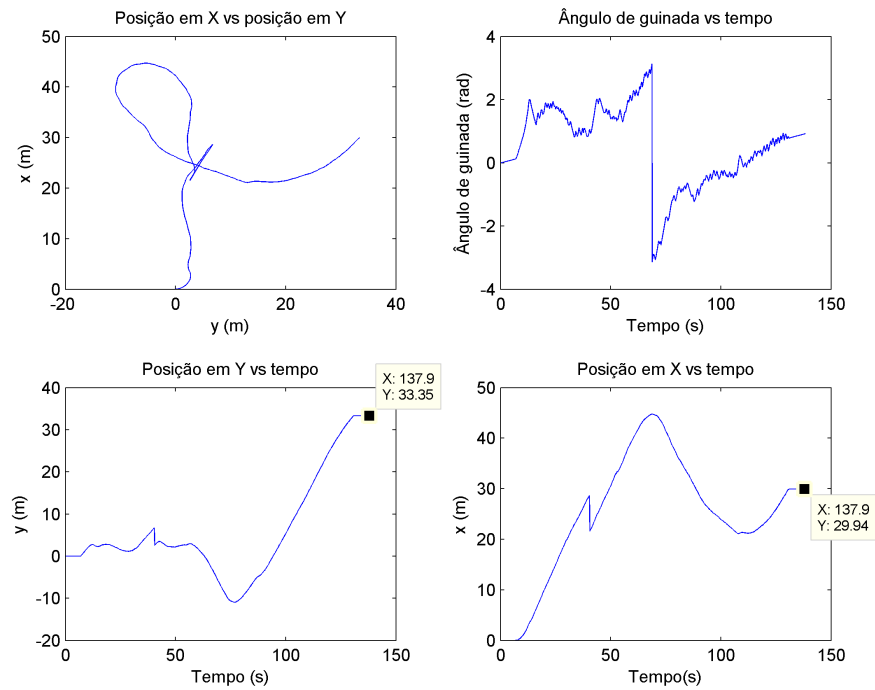


(a) Pose estimada apenas pelo modelo do processo com medidas do giroscópio

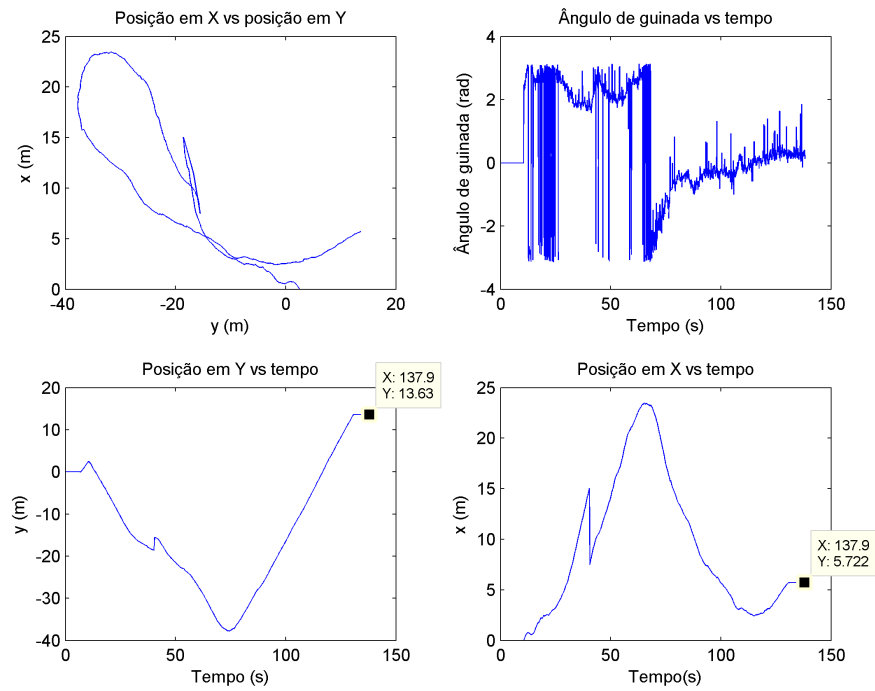


(b) Pose estimada apenas pelo modelo de medição com o algoritmo TRIAD

Figura 4.6: Estimação da pose do robô na Trajetória 2



(a) Pose estimada apenas pelo modelo do processo com medidas do giroscópio



(b) Pose estimada apenas pelo modelo de medição com o algoritmo TRIAD

Figura 4.7: Estimação da pose do robô na Trajetória 3

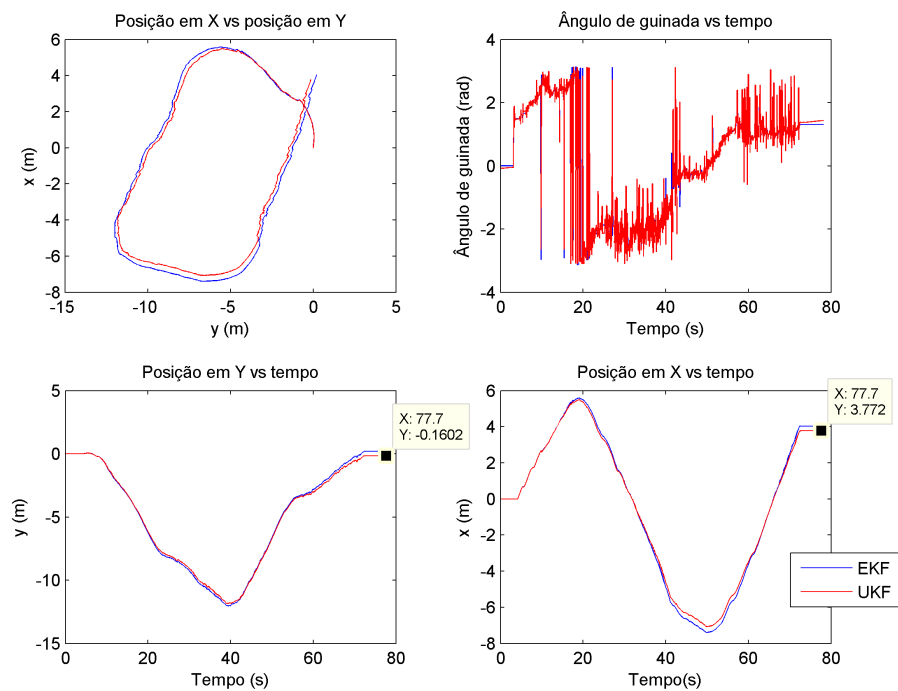


Figura 4.8: Pose estimada do robô na Trajetória 1 após filtragem

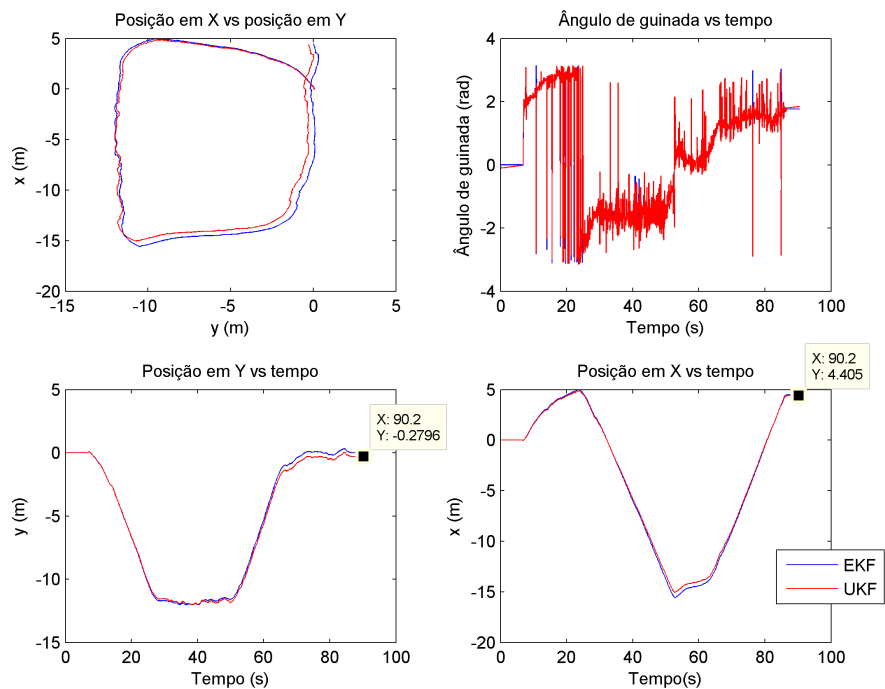


Figura 4.9: Pose estimada do robô na Trajetória 2 após filtragem

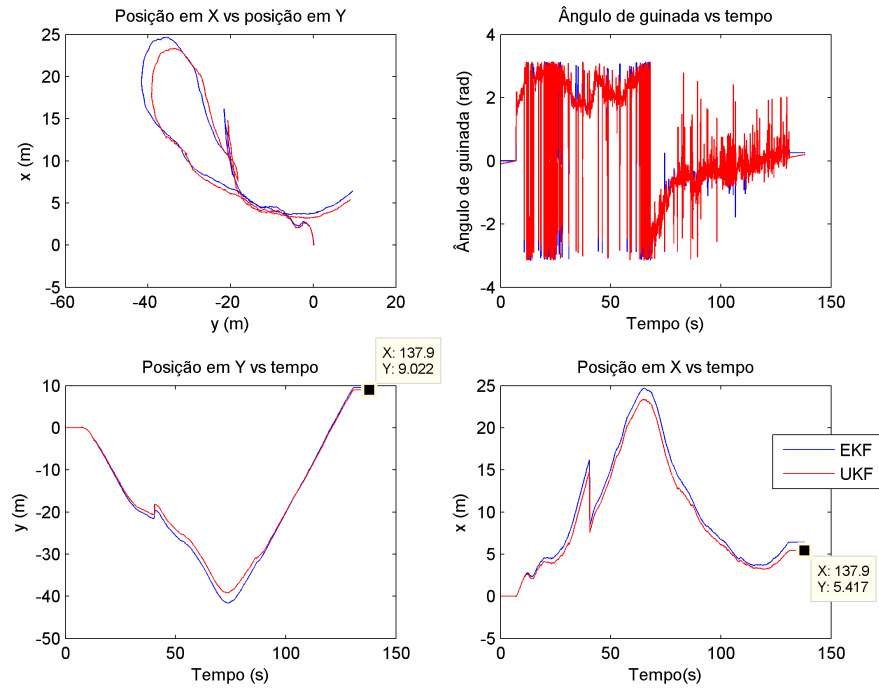


Figura 4.10: Pose estimada do robô na Trajetória 3 após filtragem

próximas. Quando o oposto ocorre, com maior confiança no modelo de processo ruidoso, aumenta-se a discrepância entre os filtros, com melhor performance do Filtro de Kalman *Unscented*. Assim, pode-se concluir que o UKF apresenta uma resposta melhor a sistemas ruidosos do que o EKF.

Um dos parâmetros para analisar a qualidade dos filtros consiste em comparar as distâncias entre o ponto inicial e final, que, no mundo real, foi próxima de zero, uma vez que o robô realizou caminhos fechados. A Tabela 4.1 mostra essa comparação. Como o resultado do EKF e do UKF foram muito próximos, não foi feita a distinção dos dois na tabela.

	Predição	Medição	EKF/UKF
Trajetória 1	7.48 m	4.54 m	3.77 m
Trajetória 2	11.45 m	8.07 m	4.41 m
Trajetória 3	44.79 m	14.78 m	10.52 m

Tabela 4.1: Distância entre os pontos inicial e final das trajetórias para cada método utilizado

A análise da Tabela 4.1 permite observar com clareza a integração dos erros da predição, que é maior quanto maior a distância percorrida. Permite observar também que o resultado após a filtragem é sempre melhor.

Assim, comparando os resultados dos filtros das Figuras 4.8 a 4.10 com as trajetórias aproximadas percorridas mostradas na Figura 4.1, percebe-se que, mesmo sem a utilização do GPS, os resultados foram bastante satisfatórios, onde os formatos e distâncias percorridas depois da filtragem se assemelham aos formatos e distâncias reais. Além disso, para o contexto da competição, o robô contará com um algoritmo de visão computacional para encontrar o cone de trânsito. Assim,

quando o robô estiver próximo do ponto de destino, esse algoritmo será incorporado ao sistema de localização, melhorando sua performance.

4.2 Teste de Localização por Odometria em Tempo Real

O teste de localização por odometria, implementada para testar o controle de movimento do robô, foi realizado com os motores do robô rodando livremente, sem carga, ou seja, com o robô suspenso sem tocar o chão. O resultado pode ser observado na Figura 4.11, em que é mostrado a trajetória realizada pelo robô, devido ao controle implementado em [13] e a trajetória de referência, calculada pelo planejamento de rotas implementado em [12].

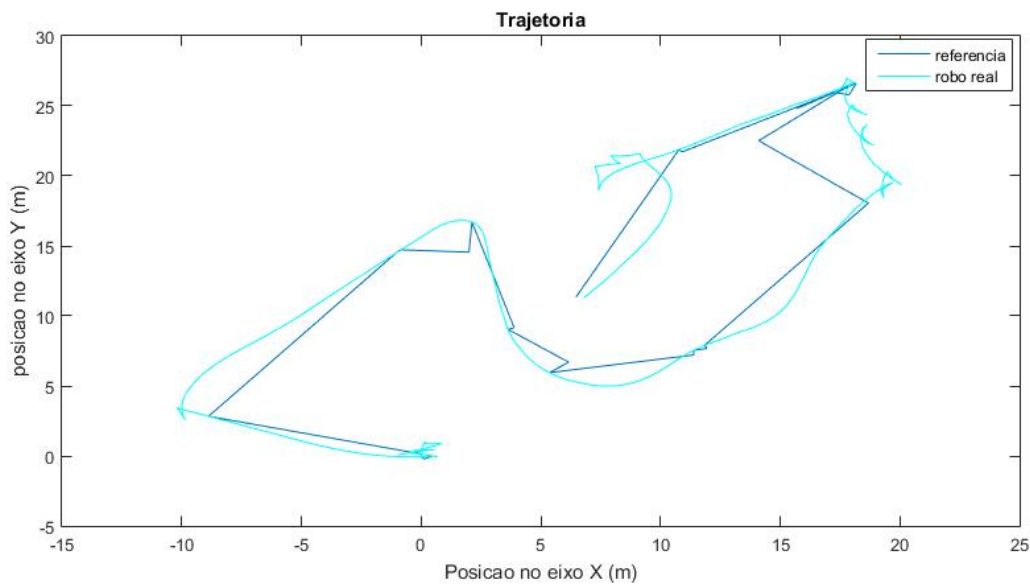


Figura 4.11: Trajetória real do robô vs trajetória de referência utilizando um sistema de localização em tempo real por odometria

4.3 Dificuldades Encontradas

Durante a realização dos experimentos, foram encontradas algumas dificuldades, que impossibilitaram completar totalmente os objetivos deste trabalho. Os principais problemas encontrados foram:

- Fragilidade da plataforma robótica. Como o robô foi inteiramente montado por alunos da equipe DROID, de forma praticamente manual, sem o uso de equipamentos adequados para construir uma plataforma robusta, era comum que o robô apresentasse problemas de *hardware* ou de eletrônica, o que atrasou muito o teste dos *softwares* desenvolvidos. Além disso, a estrutura acabou ficando muito pesada e alta para o *chassi* adquirido e o robô não era

simétrico. Dessa forma, o robô estava muito suscetível a vibrações, o que gera problemas tanto na estrutura quanto degrada as soluções fornecidas pelos sensores relativos.

- Necessidade de aquisição de um novo receptor de GPS. O primeiro GPS apresentava soluções pouco precisas e com uma taxa de amostragem muito alta, principalmente por conta de problemas com a antena. Assim, foi necessário adquirir outro receptor, de outro fabricante, com outros protocolos e especificações. Isso fez com que muito tempo fosse perdido reescrevendo o programa de coleta de dados do GPS.
- Dificuldade de teste do sistema. Por se tratar de um sistema de localização em ambientes externos baseado em GPS, a única possível de se realizar testes completos do sistema é do lado de fora do ambiente de trabalho da DROID. Programar do lado de fora tornou-se especialmente trabalhoso pois os testes foram feitos por meio do acesso remoto ao *RaspberryPi*, o que exigia a existência de uma rede e a programação em terminal, que não é agradável.

Capítulo 5

Conclusões

Este trabalho apresentou o desenvolvimento de um sistema de localização para robôs móveis em ambientes externos, em especial para um robô da equipe de robótica DROID da UnB. Esse sistema foi baseado na integração da odometria, de um receptor de GPS e de uma unidade de navegação inercial contendo giroscópio, acelerômetro e magnetômetro triaxiais. A fusão sensorial foi realizada por meio de técnicas de filtragem Bayesianas não-lineares, o Filtro de Kalman Estendido e o Filtro de Kalman *Unscented*. Apresentou-se uma visão geral sobre as técnicas de localização, bem como a fundamentação matemática envolvida no problema.

O desenvolvimento do projeto envolveu a derivação das equações dos filtros para o caso estudado, com a modelagem matemática do modelo do processo e do modelo de medição. Foi realizado um estudo detalhado dos sensores utilizados, bem como a implementação de programas para a coleta de dados desses sensores. A implementação *online* dos algoritmos de filtragem não foi possível, devido a grande ocorrência de problemas na estrutura mecânica do robô, o que dificultou muito a realização de testes no *software*. A incorporação das medidas do GPS também não foi possível, devido a um problema na comunicação do receptor com o *RaspberryPi*.

Assim, foi realizada a coleta de dados provenientes da odometria e da IMU e a filtragem *offline* desses dados por meio do EKF e do UKF, para se obter a estimativa da atitude do robô, de forma que sua posição e velocidade são estimadas apenas por odometria. Os resultados obtidos validam a abordagem adotada.

Apesar de esse trabalho ter sido desenvolvido para um robô específico, as técnicas aqui apresentadas podem ser aplicadas a diversas plataformas robóticas. Assim, espera-se que o resultado aqui apresentado possa ser utilizado pela equipe DROID e em outros projetos que envolvam navegação autônoma.

5.1 Trabalhos Futuros

Para que seja possível implementar o sistema de localização completo no robô, ainda há muito trabalho a fazer. O mais imediato consiste em fazer a leitura correta do receptor de GPS. Para

isso, propõe-se que um estudo mais aprofundado sobre a comunicação serial do *RaspberryPi* seja realizado, de forma a se conseguir trabalhar com o barramento UART sem a necessidade de bibliotecas externas. Outra abordagem para resolver esse problema é realizar as leituras do receptor no Arduino e este as passar para o *RaspberryPi*.

Outra proposta é realizar uma melhor análise do comportamento dos sensores, de forma a determinar de maneira mais correta as variâncias de cada um, melhorando o comportamento do filtro. Além disso, seria importante implementar a calibração *online* do giroscópio e do magnetômetro, com estimação de seus *biases* e fatores de escala durante a execução do filtro.

Por fim, deve-se realizar a implementação *online* dos algoritmos de localização apresentados e, posteriormente, integrá-los com o restante da estrutura de navegação.

REFERÊNCIAS BIBLIOGRÁFICAS

- [1] GROVES, P. D. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. : Artech House, 2008.
- [2] JEFFREY, C. *An Introduction to GNSS: Gps, glonass, galileo and other global navigation satellite systems*. : NovAtel Inc, 2010.
- [3] WAN, E. A.; MERWE, R. van der. The unscented kalman filter. In: HAYKIN, S. (Ed.). *Kalman Filtering and Neural Networks*. : John Wiley and Sons, 2001. cap. 7, p. 221–276.
- [4] COX, I.; WILFONG, G. (Ed.). *Autonomous Robot Vehicles*. : Springer-Verlag, 1990.
- [5] NEGENBORN, R. *Robot Localization and Kalman Filters: On finding your position in a noisy world*. Tese (dissertation) — Utrecht University, 2013.
- [6] BEKEY, G. A. *Autonomous Robots: from biological inspiraion to implemetation and control*. : The MIT Press, 2005.
- [7] COX, I. J. Blanche: Position estimation for an autonomous robot vehicle. *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, p. 432 – 439, Sep 1989.
- [8] BORENSTEIN, J. et al. Mobile robot positioning: Sensors and techniques. *Journal of Robotic Systems*, v. 14, p. 231 – 249, Apr 1997.
- [9] THRUN, S. et al. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, v. 23, p. 661 – 692, Jun 2006.
- [10] URMSON, C.; ANHALT, J.; BAGNELL, D. Autonomous driving in urbanenvironments: Boss and the urban challenge. *Journal of Field Robotics*, v. 25, p. 425 – 466, Jun 2008.
- [11] CHOSET, H. et al. *Principles of Robot Motion: Theory, algorithms, and implementations*. : The MIT Press, 2005.
- [12] OLIVEIRA, R. W. S. M. de. *Uma Arquitetura de Navegação para Robôs Móveis*. 2017. Trabalho de Graduação - Universidade de Brasília.
- [13] PORTO, L. H. S. *Controle de Movimento de um Robô Não-Holonômico com Tração Diferencial*. 2017. Trabalho de Graduação - Universidade de Brasília.

- [14] GEL, P.; ROUMELIOTIS, S. I.; SUKHATME, G. S. Robust localization using relative and absolute position estimates. *IEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 1999.
- [15] BLACK, H. D. A passive system for determining the attitude of a satellite. *AIAA Journal*, v. 2, p. 1350–1351, Jul 1964.
- [16] Bó, A. P. L. *Desenvolvimento de um Sistema de Localização 3D para Aplicação em Robôs Aéreos*. Tese (dissertation) — Universidade de Brasília, 2007.
- [17] THRUN, S.; BURGARD, W.; FOX, D. *Probabilistic Robotics*. : The MIT Press, 2005.
- [18] COSTA, A. H. R.; SELVATICI, A. H. P. Localização usando representação do ambiente. In: ROMERO, R. A. F. et al. (Ed.). *Robótica Móvel*. : LTC, 2014. cap. 8, p. 139 – 160.
- [19] KALMAN, R. E. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, v. 82, p. 35–45, Mar 1960.
- [20] DURRANT-WHYTE, J. L. . H. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, v. 7, p. 376 – 382, Jun 1991.
- [21] CROWLEY, F. C. . J. Position estimation for a mobile robot using vision and odometry. *1992 IEEE International Conference on Robotics and Automation*, Mai 1992.
- [22] DURRANT-WHYTE, B. B. . H. Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation*, v. 11, p. 328 – 342, Jun 1995.
- [23] JULIER, S.; UHLMANN, J.; DURRANT-WHYTE, H. A new approach for filtering nonlinear systems. *Proceedings of the American Control Conference*, p. 1628–1638, Jun 1995.
- [24] JULIER, S.; UHLMANN, J. A general method for approximating nonlinear transformations of probability distributions. *Technical Report, RRG, Department of Engineering Science, University of Oxford*, Nov 1996.
- [25] JULIER, S.; UHLMANN, J. A new extension of the kalman filter to nonlinear systems. *Signal Processing, Sensor Fusion, and Target Recognition*, Jul 1997.
- [26] WAN, E. A.; MERWE, R. van der. The unscented kalman filter for nonlinear estimation. *Adaptive Systems for Signal Processing, Communications, and Control Symposium*, p. 153–158, Oct 2000.
- [27] MENEGAZ, H. M. T. et al. A systematization of the unscented kalman filter theory. *IEEE Transactions on Automatic Control*, p. 2583 – 2598, Oct 2015.
- [28] CRASSIDIS, J. Unscented filtering for spacecraft attitude estimation. *Journal of Guidance Control and Dynamics*, v. 26, p. 536–542, Jul 2003.
- [29] BORGES, G. A. Mapeamento e localização simultâneos. In: ROMERO, R. A. F. et al. (Ed.). *Robótica Móvel*. : LTC, 2014. cap. 10, p. 178 – 207.

- [30] NEO-6 u-blox 6 GPS Modules Data Sheet. Disponível em: https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf.
- [31] U-BLOX 6 Receiver Description Including Protocol Specification. Disponível em: https://www.u-blox.com/sites/default/files/products/documents/u-blox6_ReceiverDescrProtSpec_%28GPS.G6-SW-10018%29_Public.pdf.
- [32] MPU-9250 Product Specification. Disponível em: <https://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf>.
- [33] MPU-9250 Register Map and Descriptions. Disponível em: <https://www.invensense.com/wp-content/uploads/2015/02/RM-MPU-9250A-00-v1.6.pdf>.
- [34] KUIPERS, J. B. (Ed.). *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*. : Princeton University Press, 2002.

ANEXOS

I. DESCRIÇÃO DO CONTEÚDO DO CD

No CD entregue com o trabalho é disponibilizada uma cópia digital do relatório e uma pasta com todos os arquivos utilizados em sua elaboração.

Os *softwares* proprietários incluem:

- Códigos em C++ de coletas de dados
- Código em C++ da calibração do acelerômetro
- Código de localização *online* por odometria
- Código em *MATLAB* da implementação do FKE e do FKU
- Dados coletados durante os testes

Visto que este trabalho faz parte de um trabalho maior de navegação autônoma, a CD também contém *softwares* de terceiros necessários para o funcionamento do sistema de localização.

O arquivo de texto *README* explica como executar os códigos.

II. FILTROS DE KALMAN

Este anexo tem como objetivo apresentar as equações dos algoritmos de fusão sensorial utilizado neste trabalho, o Filtro de Kalman Estendido e o Filtro de Kalman *Unscented*.

Como ponto de partida apresenta-se o Filtro de Kalman, que é o filtro ótimo para estimação de estados em sistemas lineares. Em seguida, discute-se sua aplicação em sistemas não-lineares através do EKF, baseado na linearização de primeira ordem das equações que descrevem o modelo do sistema, e do UKF, baseado na transformação *unscented*. As derivação das equações mostradas nesse anexo podem ser encontradas em [17] e [3].

II.1 Filtro de Kalman

O filtro de Kalman é um estimador recursivo cujo objetivo é fornecer a estimativa de mínima variância para o estado de um sistema dinâmico linear estocástico, cuja evolução segue um modelo incerto, a partir de medições ruidosas. O modelo de processo e o modelo de medição são dados por

$$\mathbf{x}_k = \mathbf{A}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k, \quad \mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k), \quad (\text{II.1a})$$

$$\mathbf{y}_k = \mathbf{C}_k \mathbf{x}_k + \mathbf{v}_k, \quad \mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k). \quad (\text{II.1b})$$

Os vetores \mathbf{x} , \mathbf{u} e \mathbf{y} são chamados de vetor de estado, vetor de controle ou entrada e vetor de medição, respectivamente. A matriz \mathbf{A} representa a transição de estados e a matriz \mathbf{B} relaciona a entrada ao estado. Os termos \mathbf{w}_k e \mathbf{v}_k representam o ruído de processo e de medição, que possuem natureza gaussiana de média nula e covariância \mathbf{Q} e \mathbf{R} .

O Filtro de Kalman é dividido em duas partes, a etapa de predição e a etapa de correção. Na fase de predição obtém-se as estimativas *a priori* do estado e da covariância do sistema, dados por

$$\hat{\mathbf{x}}_k^- = \mathbf{A}_k \hat{\mathbf{x}}_{k-1}^+ + \mathbf{B}_k \mathbf{u}_k, \quad (\text{II.2a})$$

$$\mathbf{P}_k^- = \mathbf{F}_k \mathbf{P}_{k-1}^+ \mathbf{F}_k^T + \mathbf{Q}_k. \quad (\text{II.2b})$$

Na fase de correção é calculado o ganho de Kalman que minimiza a covariância *a priori* estimada, conforme

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{C}_k^T (\mathbf{C}_k \mathbf{P}_k^- \mathbf{C}_k^T + \mathbf{R}_k)^{-1}. \quad (\text{II.3})$$

A partir daí, dada a medição \mathbf{y} , são calculadas as estimativas *a posteriori* do estado e da covariância dos sistema, dadas por

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \mathbf{C}_k \hat{\mathbf{x}}_k^-) \quad (\text{II.4a})$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k) \mathbf{P}_k^- (\mathbf{I} - \mathbf{K}_k \mathbf{C}_k)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T \quad (\text{II.4b})$$

II.2 Filtro de Kalman Estendido

O Filtro de Kalman Estendido é uma das soluções utilizadas quando os modelos de processo e de medição são não-lineares. O EKF promove uma linearização de primeira ordem do modelo em torno da estimativa atual, conforme os passos

- Modelo não-linear

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k, \quad \mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k) \quad (\text{II.5a})$$

$$\mathbf{y}_k = h(\mathbf{x}_k) + \mathbf{v}_k, \quad \mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k) \quad (\text{II.5b})$$

- Linearização do Modelo

$$\mathbf{x}_k \approx f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k) + \mathbf{F}_k (\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) \quad (\text{II.6a})$$

$$\mathbf{y}_k \approx h(\hat{\mathbf{x}}_k) + \mathbf{H}_k (\mathbf{x}_k - \hat{\mathbf{x}}_k) \quad (\text{II.6b})$$

$$\mathbf{F}_k = \frac{\partial f(\mathbf{x}_{k-1}, \mathbf{u}_k)}{\partial \mathbf{x}_{k-1}}, \quad (\text{II.6c})$$

$$\mathbf{H}_k = \frac{\partial h(\mathbf{x}_k)}{\partial \mathbf{x}_k}. \quad (\text{II.6d})$$

- Etapa de predição

$$\hat{\mathbf{x}}_k^- = f(\hat{\mathbf{x}}_{k-1}^+, \mathbf{u}_k) \quad (\text{II.7a})$$

$$\mathbf{P}_k^- = \mathbf{F}_k \mathbf{P}_{k-1}^+ \mathbf{F}_k^T + \mathbf{Q}_k \quad (\text{II.7b})$$

- Etapa de correção

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \quad (\text{II.8a})$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - h(\hat{\mathbf{x}}_k^-)) \quad (\text{II.8b})$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^- (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T \quad (\text{II.8c})$$

II.3 Filtro de Kalman Unscented

O Filtro de Kalman *Unscented* utiliza uma transformação *unscented* para estimar o estado de sistemas não-lineares sem realizar a linearização dos modelos do sistema e de medição. Essa transformação usa um conjunto de amostras (chamado de sigma-pontos) cuidadosamente escolhidas de forma determinística, que parametrizam a média e a covariância da crença. A função do sistema é aplicada a cada amostra, o que resulta em um grupo de pontos transformados. A média e a covariância desse grupo de pontos são a média e a covariância propagada da crença. O UKF trabalha com a premissa de que é mais fácil aproximar uma distribuição gaussiana por meio de um número fixo de parâmetros do que por meio da aproximação de uma função não-linear.

O UKF segue os passos abaixo, em que n representa o tamanho do vetor de estados.

- Cálculo dos sigma-pontos

$$\sigma_{k-1} = \sqrt{(n + \kappa)(\mathbf{P}_{k-1}^+ + \mathbf{Q}_k)} \quad (\text{II.9})$$

Para $i = 1, \dots, n$,

$$\chi_{k-1}(0) = \hat{\mathbf{x}}_{k-1}^+ \quad (\text{II.10a})$$

$$\chi_{k-1}(i) = \hat{\mathbf{x}}_{k-1}^+ + \sigma_{k-1} \quad (\text{II.10b})$$

$$\chi_{k-1}(i + n) = \hat{\mathbf{x}}_{k-1}^+ - \sigma_{k-1} \quad (\text{II.10c})$$

- Propagação dos sigma-pontos

Para $i = 0, \dots, 2n$,

$$\chi_k(i) = f(\chi_{k-1}(i), k) \quad (\text{II.11})$$

- Cálculo da média e da covariância da predição

$$\hat{\mathbf{x}}_k^- = \frac{1}{n + \kappa} \left\{ \kappa \chi_k(0) + \frac{1}{2} \sum_{i=1}^{2n} \chi_k(i) \right\} \quad (\text{II.12a})$$

$$\mathbf{P}_k^- = \frac{1}{n + \kappa} \left\{ \kappa [\chi_k(0) - \hat{\mathbf{x}}_k^-][\chi_k(0) - \hat{\mathbf{x}}_k^-]^T + \frac{1}{2} \sum_{i=1}^{2n} [\chi_k(i) - \hat{\mathbf{x}}_k^-][\chi_k(i) - \hat{\mathbf{x}}_k^-]^T \right\} + \mathbf{Q}_k \quad (\text{II.12b})$$

- Cálculo dos sigma-pontos da predição da medição

Para $i = 0, \dots, 2n$,

$$\gamma_k(i) = h(\chi_k(i), k) \quad (\text{II.13})$$

- Cálculo das estatísticas da predição da medição

$$\hat{\mathbf{y}}_k = \frac{1}{n + \kappa} \left\{ \kappa \gamma_k(0) + \frac{1}{2} \sum_{i=1}^{2n} \gamma_k(i) \right\} \quad (\text{II.14a})$$

$$\mathbf{P}_k^{yy} = \frac{1}{n + \kappa} \left\{ \kappa [\gamma_k(0) - \hat{\mathbf{y}}_k][\gamma_k(0) - \hat{\mathbf{y}}_k]^T + \frac{1}{2} \sum_{i=1}^{2n} [\gamma_k(i) - \hat{\mathbf{y}}_k][\gamma_k(i) - \hat{\mathbf{y}}_k]^T \right\} + \mathbf{R}_k \quad (\text{II.14b})$$

$$\mathbf{P}_k^{xy} = \frac{1}{n + \kappa} \left\{ \kappa [\chi_k(0) - \hat{\mathbf{x}}_k^-][\gamma_k(0) - \hat{\mathbf{y}}_k]^T + \frac{1}{2} \sum_{i=1}^{2n} [\chi_k(i) - \hat{\mathbf{x}}_k^-][\gamma_k(i) - \hat{\mathbf{y}}_k]^T \right\} \quad (\text{II.14c})$$

- Etapa de correção

$$\mathbf{K}_k = \mathbf{P}_k^{xy} (\mathbf{P}_k^{yy})^{-1} \quad (\text{II.15a})$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \hat{\mathbf{y}}_k) \quad (\text{II.15b})$$

$$\mathbf{P}_k^+ = \mathbf{P}_k^- - \mathbf{K}_k \mathbf{P}_k^{yy} \mathbf{K}_k^T \quad (\text{II.15c})$$

II.3.1 Algoritmo USQUE

O algoritmo USQUE foi proposto para estimar a atitude com quatérnios usando a transformação *unscented*. Nessa abordagem, em vez de se utilizar diretamente o quatérnio, utiliza-se o erro multiplicativo do quatérnio, através de um vetor de erro de três componentes, eliminando as restrições. A representação escolhida para o vetor de erro foi a de parâmetros generalizados de Rodrigues.

O quatérnio de erro é dado por

$$\delta \mathbf{q} = [\delta q_0 \quad \delta \mathbf{Q}^T]^T. \quad (\text{II.16})$$

O vetor de erro representado por parâmetros de Rodrigues generalizado é dado por

$$\delta \mathbf{d} = l \frac{\delta \mathbf{Q}}{a + \delta q_0}. \quad (\text{II.17})$$

A transformação do vetor de erro para quatérnio de erro é dado por

$$\delta q_0 = \frac{-a \|\delta \mathbf{d}\|^2 + l \sqrt{l^2 + (1 - a^2) \|\delta \mathbf{d}\|^2}}{l^2 + \|\delta \mathbf{d}\|^2}, \quad (\text{II.18a})$$

$$\delta \mathbf{Q} = l^{-1} (a + \delta q_0) \delta \mathbf{d}, \quad (\text{II.18b})$$

em que $0 < a < 1$ e $l = 2(a + 1)$.

O algoritmo USQUE segue as mesmas etapas do UKF apresentado, acrescentando as transformações entre as formas de representação do quatérnio, conforme os passos abaixo.

- Cálculo dos sigma-pontos

Cálculo dos sigma-pontos do vetor de erro

Para $i = 1, \dots, n$,

$$\chi_{k-1}^{\delta \mathbf{d}}(0) = [\mathbf{0}]_{3 \times 1} \quad (\text{II.19a})$$

$$\chi_{k-1}^{\delta \mathbf{d}}(i) = [\mathbf{0}]_{3 \times 1} + \boldsymbol{\sigma}_{k-1}(i) \quad (\text{II.19b})$$

$$\chi_{k-1}^{\delta \mathbf{d}}(i+n) = [\mathbf{0}]_{3 \times 1} - \boldsymbol{\sigma}_{k-1}(i) \quad (\text{II.19c})$$

Cálculo dos sigma-pontos do quatérnio de erro

$$\chi_{k-1}^{\mathbf{q}}(0) = \hat{\mathbf{q}}_{k-1}^+ \quad (\text{II.20a})$$

$$\chi_{k-1}^{\mathbf{q}}(i) = \chi_{k-1}^{\delta \mathbf{q}}(i) \otimes \hat{\mathbf{q}}_{k-1}^+ \quad (\text{II.20b})$$

Cálculo dos sigma-pontos do quatérnio nominal

$$\chi_k^{\delta \mathbf{q}}(i) = \chi_k^{\mathbf{q}}(i) \otimes [\hat{\mathbf{q}}_{k-1}^+]^{-1} \quad (\text{II.21})$$

- Propagação dos sigma-pontos

Propagação dos sigma-pontos do quatérnio nominal

$$\chi_k^{\mathbf{q}} = f(\chi_{k-1}^{\mathbf{q}}) \quad (\text{II.22})$$

Propagação dos sigma-pontos do quatérnio de erro

Para $i = 0, \dots, 2n$,

$$\chi_k^{\delta \mathbf{q}} = \chi_k^{\mathbf{q}} \otimes [\hat{\mathbf{x}}_{k-1}^{\mathbf{q},+}]^{-1} \quad (\text{II.23})$$

Propagação dos sigma-pontos do vetor de erro

Para $i = 1, \dots, 2n$,

$$\chi_k^{\delta \mathbf{d}}(0) = [0]_{3 \times 3} \quad (\text{II.24a})$$

$$\chi_k^{\delta \mathbf{d}}(i) = l \frac{\chi_k^{\delta \mathbf{Q}}(i)}{a + \chi_k^{\delta q_0}(i)} \quad (\text{II.24b})$$

- Cálculo da média e covariância da predição, conforme Equação II.12.
- Cálculo dos sigma-pontos da predição da medição, conforme Equação II.13.

- Cálculo das estatísticas da predição da medição, conforme Equação II.14.
- Etapa de correção, conforme Equação II.15.

Cálculo da estimação corrigida do quatérnio de erro

$$\hat{\mathbf{x}}_k^{\delta q_0} = \frac{-a \left\| \hat{\mathbf{x}}_k^{\delta \mathbf{d}} \right\|^2 + l \sqrt{l^2 + (1 - a^2) \left\| \hat{\mathbf{x}}_k^{\delta \mathbf{d}} \right\|^2}}{l^2 + \left\| \hat{\mathbf{x}}_k^{\delta \mathbf{d}} \right\|^2}, \quad (\text{II.25a})$$

$$\hat{\mathbf{x}}_k^{\delta \mathbf{Q}} = l^{-1}(a + \delta q_0)\delta \mathbf{d}, \quad (\text{II.25b})$$

Cálculo da estimação corrigido quatérnio unitário

$$\hat{\mathbf{x}}_k^{\mathbf{q},+} = \hat{\mathbf{x}}_k^{\delta \mathbf{q}} \otimes \hat{\mathbf{x}}_k^{\mathbf{q},-} \quad (\text{II.26})$$

III. QUATERNIOS

Este anexo tem como objetivo descrever a utilização de quatérnios para representação de atitude e sequências de rotações. Apresenta-se algumas propriedades básicas de quatérnios, bem como as equações de conversão entre ângulos de Euler e quatérnios. Maiores detalhes sobre diferentes formas de representação e suas derivações podem ser encontradas em [34].

III.1 Definição

Os quatérnios foram criados por Sir W. Hamilton em 1943 e podem ser vistos como uma extensão do conjunto dos números complexos em que a parte complexa contém três elementos imaginários, definidos como

$$\mathbf{q} = q_0 + q_1 i + q_2 j + q_3 k = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}, \quad (\text{III.1})$$

em que \mathbf{i} , \mathbf{j} e \mathbf{k} são as unidades imaginárias, que definem a parte vetorial do quatérnio. A relação que caracteriza essas unidades são definidas como

$$i^2 = j^2 = k^2 = ijk = -1. \quad (\text{III.2})$$

III.2 Propriedades

- Soma de quatérnios

$$\mathbf{q} + \mathbf{p} = \begin{bmatrix} q_0 + p_0 \\ q_1 + p_1 \\ q_2 + p_2 \\ q_3 + p_3 \end{bmatrix} \quad (\text{III.3})$$

- Multiplicação por escalar

$$a\mathbf{q} = \begin{bmatrix} aq_0 \\ aq_1 \\ aq_2 \\ aq_3 \end{bmatrix} \quad (\text{III.4})$$

- Produto de quatérnios

$$\mathbf{q} \otimes \mathbf{p} = \begin{bmatrix} q_0 p_0 - q_1 p_1 - q_2 p_2 - q_3 p_3 \\ q_0 p_1 + q_1 p_0 + q_2 p_3 - q_3 p_2 \\ q_0 p_2 - q_1 p_3 + q_2 p_0 + q_3 p_1 \\ q_0 p_3 + q_1 p_2 - q_2 p_1 + q_3 p_0 \end{bmatrix} \quad (\text{III.5})$$

- Conjugado

$$\mathbf{q}^* = \begin{bmatrix} q_0 \\ -q_1 \\ -q_2 \\ -q_3 \end{bmatrix} \quad (\text{III.6})$$

- Norma

$$\|\mathbf{q}\| = \sqrt{\mathbf{q} \otimes \mathbf{q}^*} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \quad (\text{III.7})$$

- Inversão

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{\|\mathbf{q}\|} \quad (\text{III.8})$$

- Representação matricial

$$Q(\mathbf{q}) = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & q_3 & -q_2 \\ q_2 & -q_3 & q_0 & q_1 \\ q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \quad (\text{III.9})$$

III.3 Representação de Atitude com Quatérnios

Quatérnios unitários, em que $\|\mathbf{q}\| = 1$, podem ser usados para representar atitudes e rotações.

Na representação de rotação, o eixo de rotação é definido pelos três elementos vetoriais do quatérnio, enquanto o elemento real representa a magnitude dessa rotação. O quatérnio unitário pode ser representado da seguinte forma

$$\mathbf{q} = \cos(\theta) + \mathbf{u}\sin(\theta), \quad (\text{III.10})$$

em que \mathbf{u} define o eixo de rotação e $\theta/2$ representa sua magnitude. Dessa forma, o quatérnio pode representar um vetor de rotação como

$$\mathbf{q} = \begin{bmatrix} \cos(\theta/2) \\ r_1 \sin(\theta/2) \\ r_2 \sin(\theta/2) \\ r_3 \sin(\theta/2) \end{bmatrix}. \quad (\text{III.11})$$

A transformação de um vetor no sistema M para o sistema F pode ser dado pela relação

$$\mathbf{p}^f = \mathbf{q}_f^m \otimes \mathbf{p}^m \otimes \mathbf{q}_f^{m*} \quad (\text{III.12})$$

e a transformação inversa pela relação

$$\mathbf{p}^m = \mathbf{q}_f^{m*} \otimes \mathbf{p}^f \otimes \mathbf{q}_f^m. \quad (\text{III.13})$$

A partir da Equação III.12, pode-se obter a matriz de rotação equivalente

$$\mathbf{p}^f = \mathbf{R}_f^m \mathbf{p}^m = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \mathbf{p}^m \quad (\text{III.14})$$

A Equação III.14 define uma maneira de se obter a matriz de rotação a partir do quatérnio. Também é possível obter o quatérnio a partir da matriz de rotação

$$q_0 = \frac{\sqrt{1 + R_{1,1} + R_{2,2} + R_{3,3}}}{2} \quad (\text{III.15a})$$

$$q_1 = \frac{R_{3,2} - R_{2,3}}{4q_0} \quad (\text{III.15b})$$

$$q_2 = \frac{R_{1,3} - R_{3,1}}{4q_0} \quad (\text{III.15c})$$

$$q_3 = \frac{R_{2,1} - R_{1,2}}{4q_0} \quad (\text{III.15d})$$

Assim como ocorre com as matrizes e rotação, um quatérnio também pode ser usado para representar rotações sucessivas, por meio de

$$\mathbf{q}_n = \mathbf{q}_0 \otimes \mathbf{q}_1^0 \otimes \mathbf{q}_2^1 \otimes \dots \otimes \mathbf{q}_{n-1}^{n-2} \otimes \mathbf{q}_n^{n-1} \quad (\text{III.16})$$

III.4 Propagação de Atitude

A propagação de atitude com quatérnios se dá por

$$\dot{\mathbf{q}}_f^m = \frac{1}{2} \mathbf{q}_f^m \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}_{fm}^m \end{bmatrix}, \quad (\text{III.17})$$

ou, alternativamente por

$$\dot{\mathbf{q}} = -\frac{1}{2} \mathbf{W} \mathbf{q} = \begin{bmatrix} 0 & \omega_x & \omega_y & \omega_z \\ -\omega_x & 0 & -\omega_z & \omega_y \\ -\omega_y & \omega_z & 0 & -\omega_x \\ -\omega_z & -\omega_y & \omega_x & 0 \end{bmatrix} \mathbf{q} \quad (\text{III.18})$$

A solução numérica da Equação III.18 é dada por

$$\mathbf{q}_k = e^{-\frac{1}{2} \mathbf{W} \Delta t} \mathbf{q}_{k-1} \quad (\text{III.19})$$

III.5 Transformação entre Ângulo de Euler e Quatérnio

- Ângulo de Euler para quatérnio

$$\mathbf{q}_f^m(\mathbf{e}_f^m) = \begin{bmatrix} \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) - \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) + \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) - \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) + \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) \end{bmatrix} \quad (\text{III.20})$$

- Quatérnio para ângulo de Euler

$$\mathbf{e}_f^m(\mathbf{q}_f^m) = \begin{bmatrix} \text{atan2}(2q_0q_1 + 2q_2q_3, 1 - 2(q_1^2 - q_2^2)) \\ \text{asin}(2q_0q_2 - 2q_3q_1) \\ \text{atan2}(2q_0q_3 + 2q_1q_2, 1 - 2(q_2^2 - q_3^2)) \end{bmatrix} \quad (\text{III.21})$$